9010 电机控制卡

使用手册

2010/05/01 V1.0

版权声明

运动中的机器有危险！使用者有责任在机器中设计有效的出错处理和安全保护机制，东方嘉宏没有义务或责任对由此造成的附带的或相应产生的损失负责。

前言

使用手册的用途

本手册记录了 9010 控制卡的控制功能、函数的用法、特定控制功能的编程实现等。通过阅读本手册用户可以根据自己特定的控制系统，编制用户应用程序，实现控制要求。

使用手册的使用对象

本编程手册适用于具有 c 语言编程基础或 Windows 环境下使用动态链接库的基础，同时具有一定运动控制工作经验，对伺服和步进控制的基本结构有一定了解的工程开发人员。

使用手册的主要内容

本手册分为三章：第一章《硬件指南》主要介绍 9010 运动控制卡的硬件结构，方便工程技术人员安装、调试参考；第二章《软件指南》主要介绍 9010 运动控制卡的控制功能、库函数详解及编程实现。在随产品配套的用户光盘中有相关编程示例；第三章《库函数详解》详细介绍所有库函数的格式及功能，方便工程开发人员查阅。另外，在使用手册最后附录为一些特殊功能的解释，供用户参考。

产品更多信息

公司网址是http://www.dfjzh.com 。在公司主页上有关于产品的所有信息，包括：产品介绍、技术支持、最新产品发布、演示版软件等。

同时也可以拨打电话（010－62325085）咨询关于公司和产品的更多信息。

技术支持与售后服务

你可以通过以下途径获得我们的技术支持和售后服务：

* 电子邮件：dfjzh@263.net
* 电话：（010）62325085－810
* 发函至：北京市海淀区北四环中路 209 号健翔园 1 号楼 102 室

北京东方嘉宏机电技术有限责任公司

邮编：100083

第一章 硬件指南

§1.1 概述

北京东方嘉宏公司生产的 9010/9011 系列控制卡，是基于 USB2.0 接口、硬件核心为 DSP+FPGA 架构的高性能电机控制卡，由于 USB 接口固有的安装方便性，它可广泛应用于数控机床、工业机器人、民用雕刻机、木工机械、印刷机械、装配生产线及其他产业机械等广泛领域。

§1.2 9010 规格表

9010 电机控制卡规格表

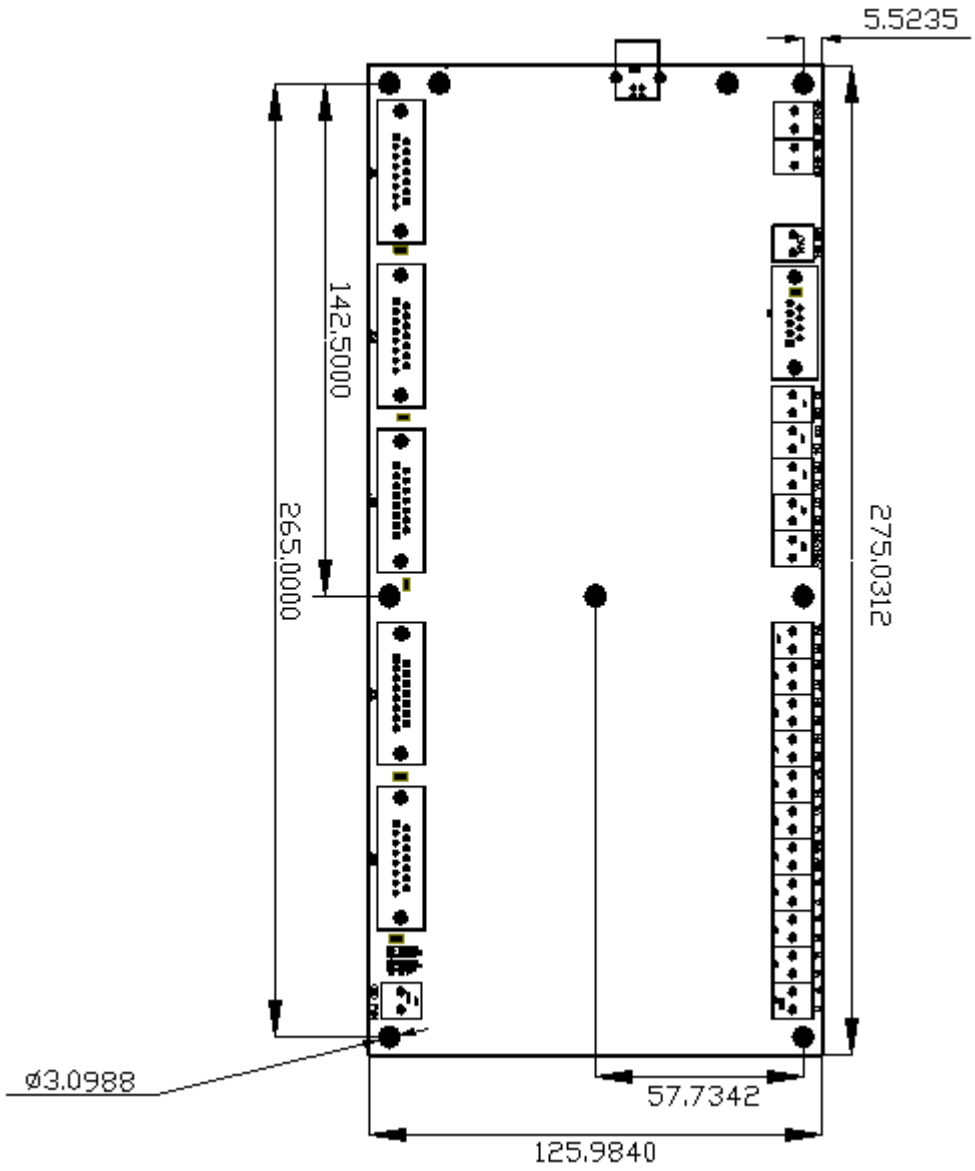
硬件指标:

总线形式	USB2.0		
控制轴数	4 轴		
轴控制输出模式	脉冲/模拟量 可设定	最高脉冲输出频率: 4Mhz 差分输出	模拟量输出范围: -10~+10V 输出精度: 不小于 13 位
编码器输入	4 路四倍频增量式	最高频率: 12.5Mhz	
辅助编码器输入	1 路四倍频增量式	最高频率: 12.5Mhz	
PWM/模拟量 辅助输出端口	1 路 PWM/模拟量 可设定	PWM 输出: 1Hz ~ 1MHz PWM 分辨率不小于 1% 模拟量: -10~+10V 输出精度: 不小于 13 位	PWM 输出: 可控制激光能量 模拟量输出: 可控制主轴转速
位置闭环调节	200 微妙 PID+速度前馈		
插补周期	500 微秒		
IO 量输出点	8+4, 12 路	8 路达灵顿 4 路光电开关	
IO 量输入点	27 路	全部光电隔离	
扩展功能	1 路 CAN 总线接口	最高可扩展 512IO 点	

软件功能

单轴控制功能			
	单轴点位控制		
	单轴速度控制		
	单轴回 HOME 点		
多轴控制功能			
	多轴电子齿轮功能		
	编码器跟随功能		
多轴插补功能			
	4 轴线性/圆弧插补	4 轴直线插补	任意 2 圆弧+2 轴直线插补
	3 轴/4 轴 NURBS 样条曲线插补		
	插补模式的编码器跟随功能	可实现硬攻丝	
	先进的插补预处理功能	内部 64 行插补数据缓冲区	自动完成插补速度规划
	插补速度调节/随时暂停	0-160%随时调节	1%精度调节
用户开发环境	Windows Xp 以上系统	VB、VC、BCB、Delphi 四种语言开发库	
WinCNC 支持	我公司功能强大的数控系统软件		

9010 控制卡主板外形尺寸



§1.3 控制卡的安装

完整的 9010 卡它包含以下内容：

① 9010 主板一块，USB 电缆一根

②光盘一张，9010 使用手册一本

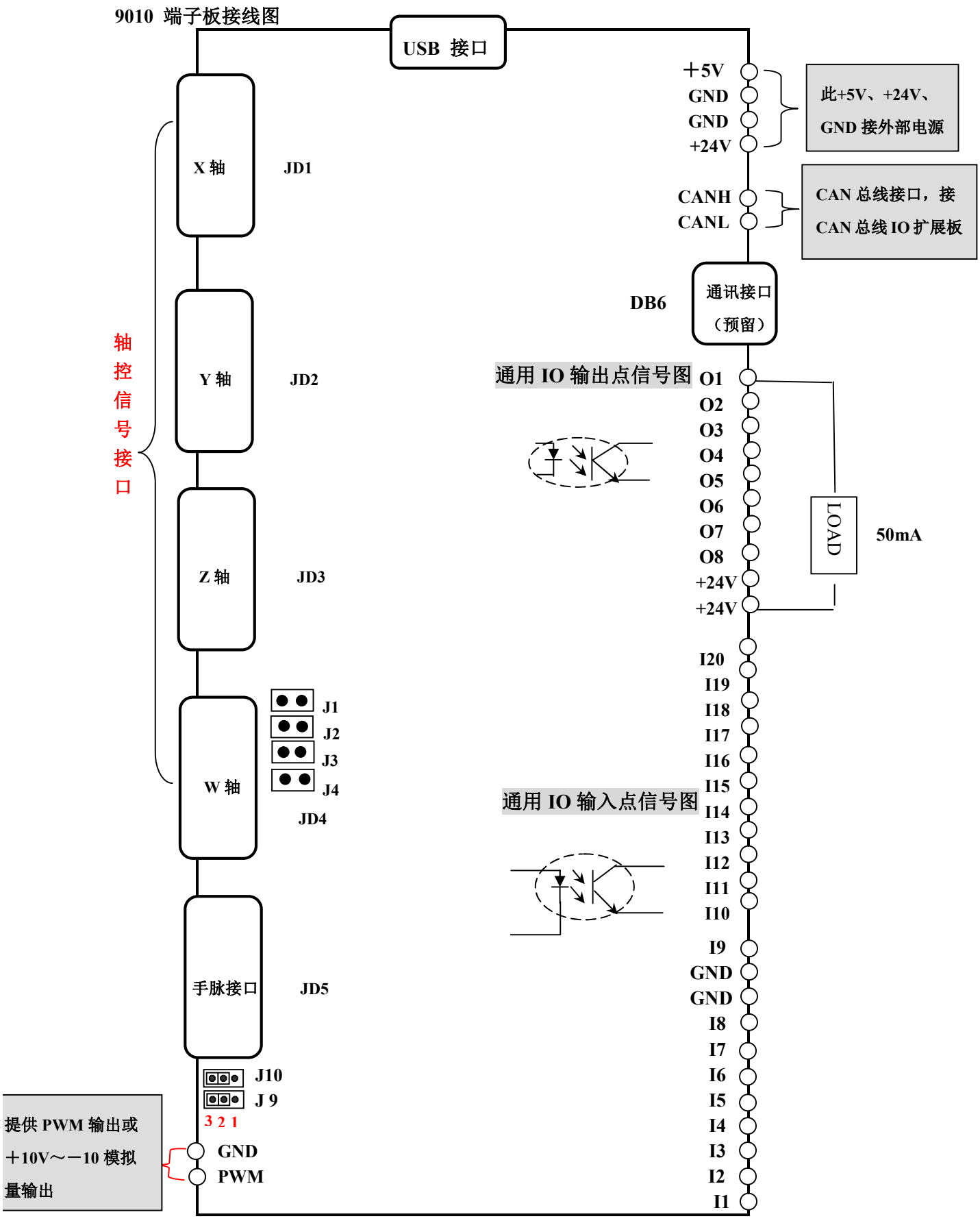
③其它配件（插头、短路棒等）

在安装控制卡前，请确保计算机 USB 接口为 2.0 版本。通过 USB2.0 电缆连接 9010 卡与 PC 机，安装 9010 的驱动程序。9010 卡的驱动程序支持 Win9x、Win2000、WinXp、WinVista，在安装驱动程序时请在光盘的 9010\driver 文件夹中寻找。

§1.4 控制卡的硬件连线

9010 卡通过端子板展开其硬件连线，在本节中将着重介绍有关 9010 端子板硬件连线原理。

9010 端子板接线图


USB 接口
+5V
GND
GND
+24V
此+5V、+24V、
GND 接外部电源
CANH
CANL
CAN 总线接口，接
CAN 总线 IO 扩展板
DB6
通讯接口
（预留）
通用 IO 输出点信号图
O1
O2
O3
O4
O5
O6
O7
O8
+24V
+24V
LOAD
50mA
I20
I19
I18
I17
I16
I15
I14
I13
I12
I11
I10
I9
GND
GND
I8
I7
I6
I5
I4
I3
I2
I1
通用 IO 输入点信号图
轴控信号接口
X 轴
JD1
Y 轴
JD2
Z 轴
JD3
W 轴
J1
J2
J3
J4
JD4
手脉接口
JD5
J10
J 9
3 2 1
GND
PWM
提供 PWM 输出或
+10V～－10 模拟
量输出

在上图中：

＊+5V 接外部电源＋5V 端

＊+24V 外接 24V 直流电源

＊ JD1－JD4 为轴控信号接口

＊ JD5 为手摇脉冲发生器接口

＊ DB6 是预留通讯接口

＊ J1、J2、J3、J4 跳线分别控制 X、Y、Z、W 轴。通过改变跳线可使 9010 卡各轴在开环控制和闭环控制之间切换

＊ J9、J10 跳线可改变输出方式为 PWM 输出或 AD 模拟量输出

＊ PWM/GND 提供一路 PWM 信号输出或-10V ～ +10V 模拟量输出

＊ I1-I20 是 20 个通用 IO 输入点

＊ O1-O8 是 8 个通用 IO 输出点

如果用户要用到 20 个通用 IO 输入点或 8 个通用 IO 输出点，必须在 GND 与+24V 之间外接 24V 电源。

§1.5 电机控制信号

9010 卡各轴可分别通过两种方式控制电机运动：开环控制(脉冲输出)、闭环控制(模拟量输出)。各轴控制方式选择可通过跳线 J1-J4 来改变，其中 J1 控制 X 轴、J2 控制 Y 轴、J3 控制 Z 轴、J4 控制 W 轴。

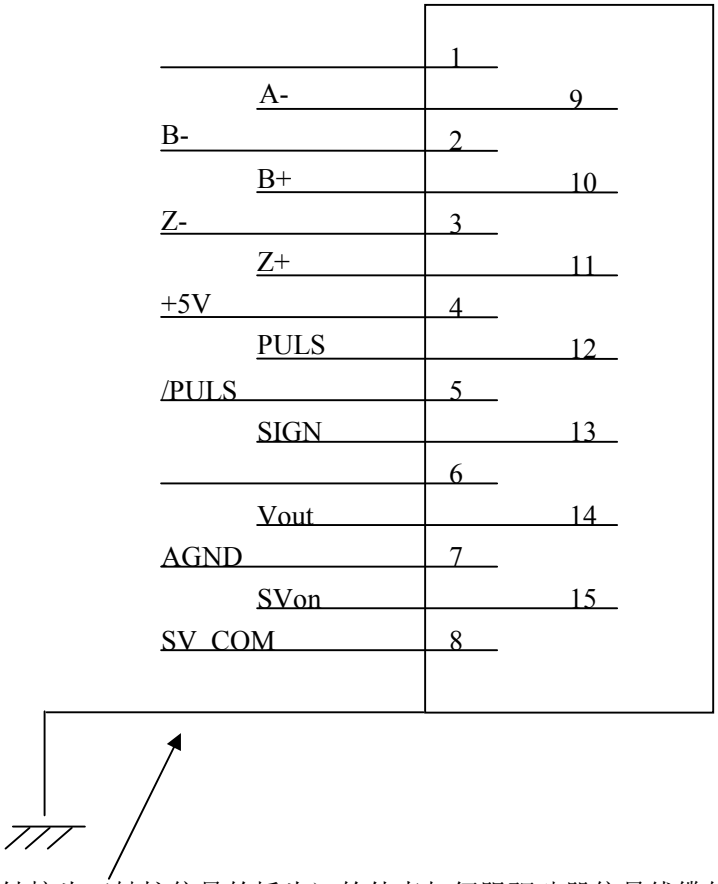
以 X 轴为例：
闭环控制时 J1 跳线(跨接)如下：

注：当 9010 卡以闭环模式工作时，轴控信号接口中脉冲输出信号 PULS、/PULS、SIGN、/SIGN **无效，见下表 （轴工作模式=5 时除外）**

开环控制时 J1 跳线(不跨接)如下：

注：当 9010 卡以开环模式工作时，轴控信号接口中模拟量输出信号 Vout、AGND **无效，见下表**

JD1－JD4 轴控信号接口（15 针）说明如下：



15 针接头（轴控信号的插头）的外壳与伺服驱动器信号线缆的屏蔽层有效连接

图 1.3.1 轴控信号接口图

表 1.

管腿名称	定义	解释
A-	编码器输入信号	
A+	编码器输入信号	
B-	编码器输入信号	
B+	编码器输入信号	
Z-	编码器输入信号	
Z+	编码器输入信号	
+5V	+5V 电源	
PLUS	轴差分脉冲输出＋	开环脉冲输出信号。 当轴以闭环模式控制时，该组信号无效 **（轴工作模式=5 时除外）**
/PLUS	轴差分脉冲输出－	
SIGN	轴差分方向输出＋	
/SIGN	轴差分方向输出－	
Vout	轴模拟量输出 ± 10V	闭环模拟量输出信号。 当轴以开环模式工作时，该组信号**无效**
AGND	轴模拟量输出模拟地	

SVon	伺服使能信号＋	
SV COM	伺服使能信号－	
外壳	15 针接头外壳(GND)	15 针接头（轴控信号的接头）的外壳与伺服驱动器信号线缆的屏蔽层有效连接

§1.5.1 伺服编码器输入连接方法

1. 伺服编码器输出与 9010 卡编码器接口一般连接方式,如下图：

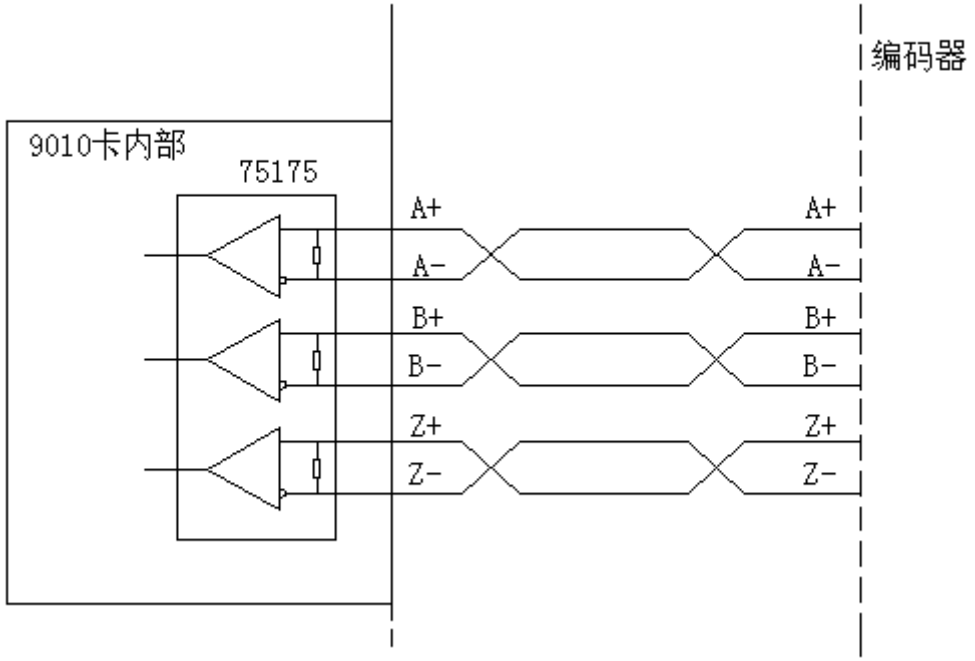
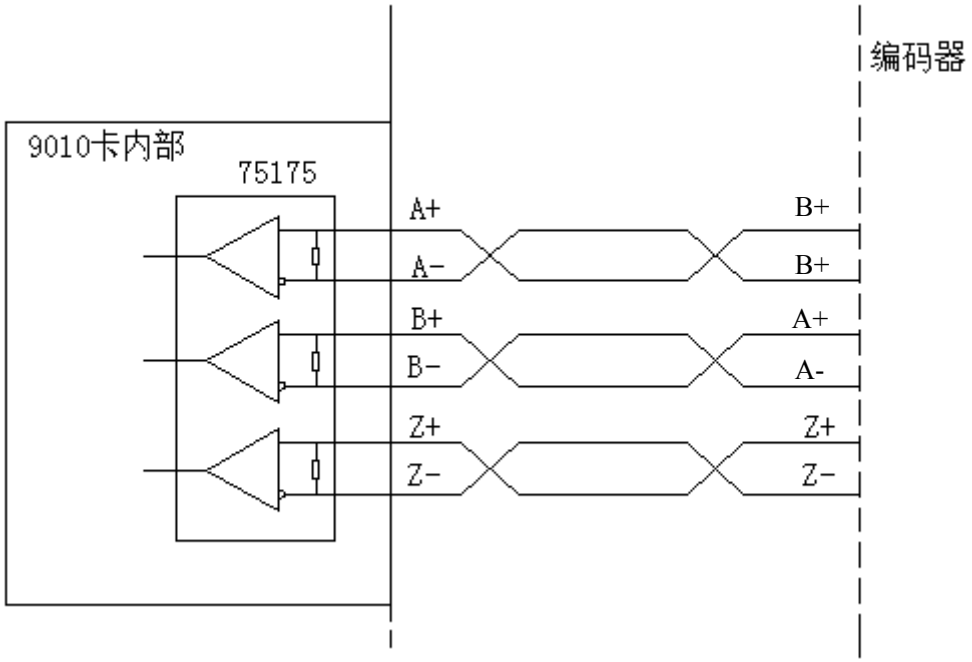


图 1.3.2 编码器差分输入信号连接图

2. 闭环系统要求 9010 编码器计数与模拟量输出成正比例，如果 9010 卡输出正电压而编码器计数反向增加，则需要改变编码器计数方向。有两种方法来实现：

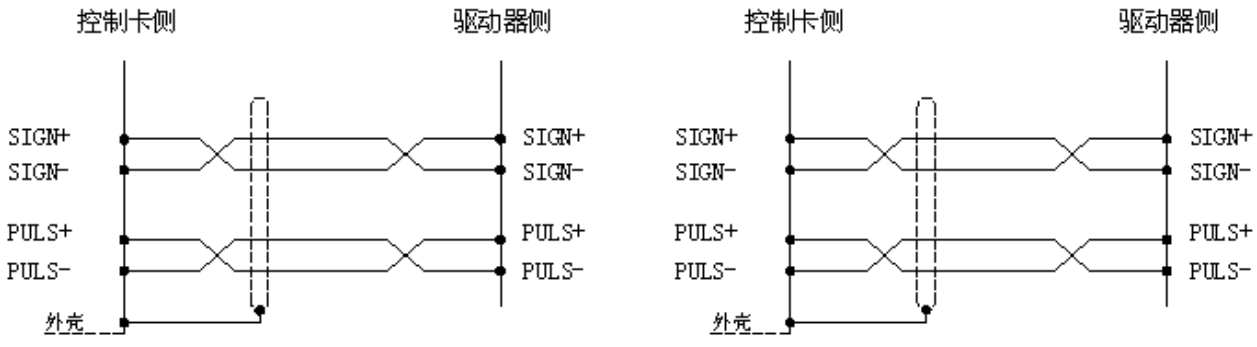
a. 若是伺服驱动器，可以设置伺服参数“反馈脉冲逻辑取反”。

b. 若没有相应的参数可以设置，则需要改变硬件接线。接线方式如下图：



§1.5.2 轴控制信号连接原理

⑴ 脉冲输出连接方法



脉冲/方向差分方式　　正脉冲/反脉冲差分方式

图 1.3.3 脉冲量控制输出信号连接图

⑵模拟量输出连接方法

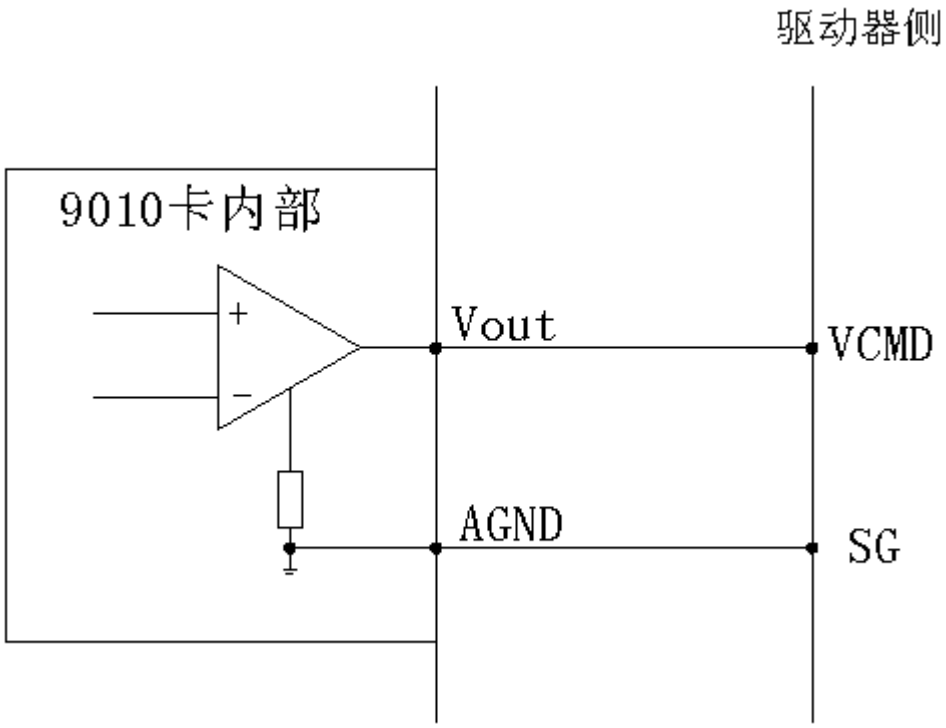


图 1.3.4 *模拟量输出模式的电气连接图*

§1.5.3 伺服使能信号连接原理

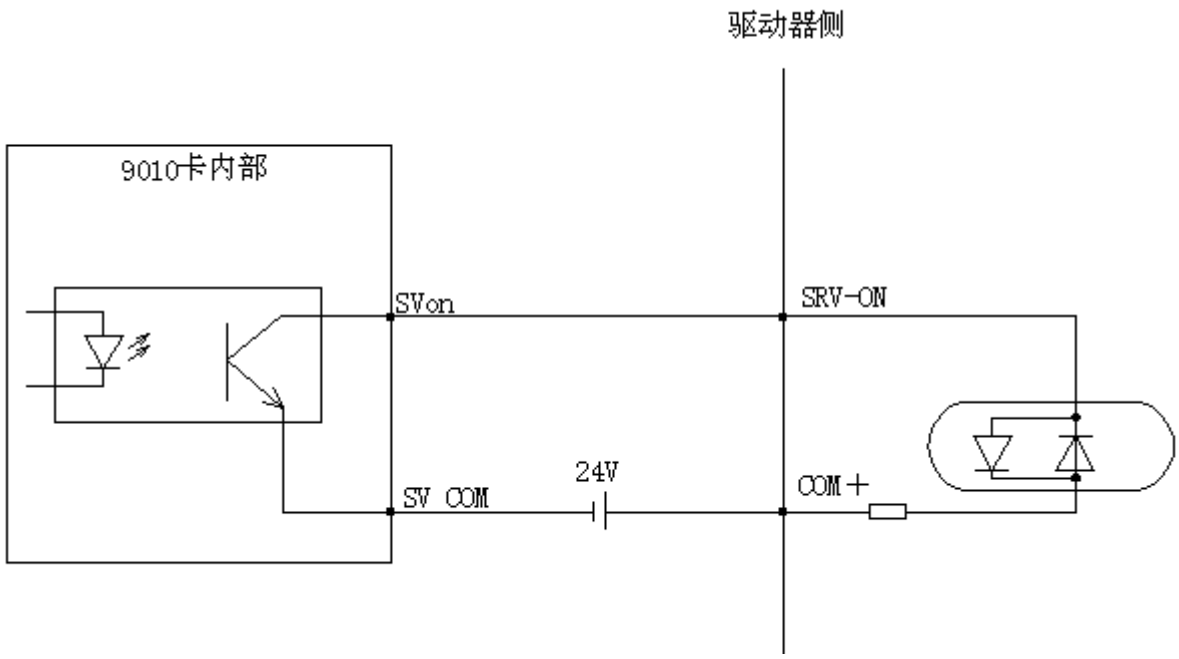
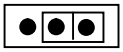


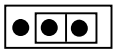
图 1.3.5 *伺服使能信号连接图*

§1.6 Vout/PWM 输出选择

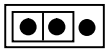
9010 有一路 PWM 或 DA 模拟量输出，通过端子板上的跳线 J9、J10 来选择是 PWM 输出还是 DA 输出。如下：

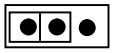
DA 输出时跳线 J10、J9 （标号 1、2 跨接）如下：


J10


J 9

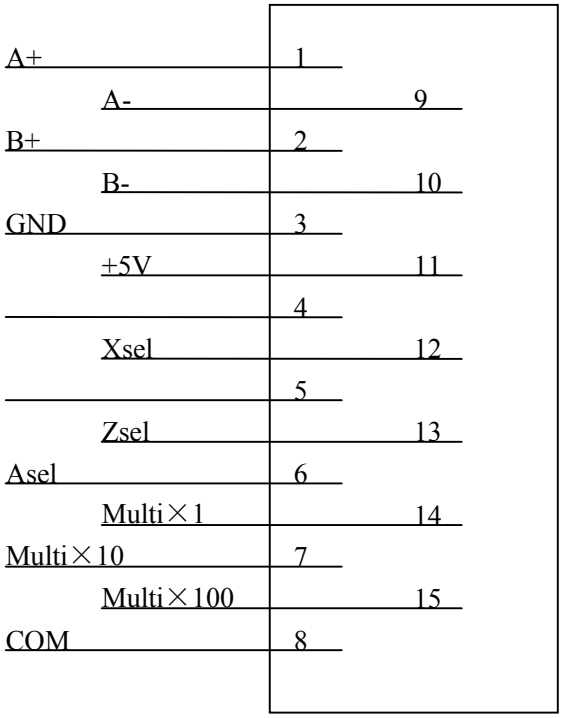
PWM输出时跳线J10、J9 （标号3、2跨接）如下：


J10


J 9

§1.7 附加编码器接口

15 针接口示意图：



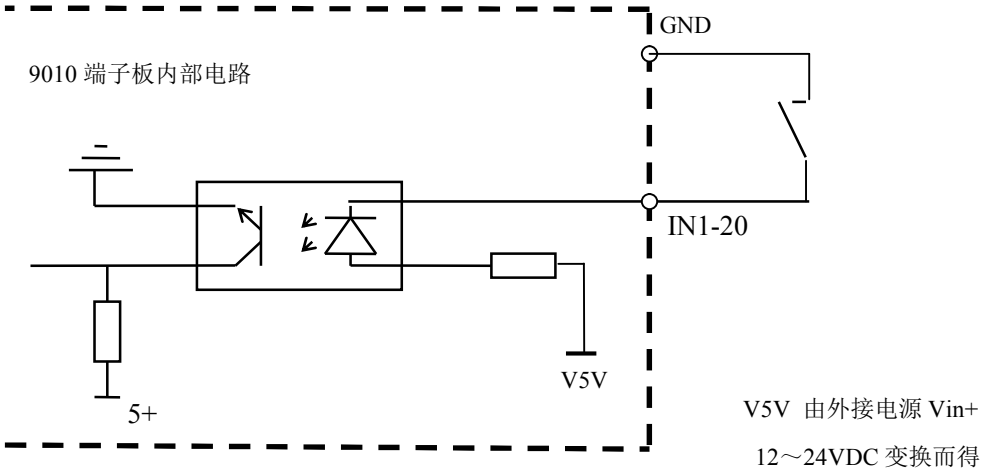
端口定义如下：

管腿名称	定义	解释
1	A+	接编码器 A+
9	A-	接编码器 A-　（编码器单端输入 可不接）
2	B+	接编码器 B+
10	B-	接编码器 B-　（编码器单端输入 可不接）
11	+5V	+5V 电源
3	GND	电源地
12	Xsel	X 轴选，　MPG_I1
5	Ysel	Y 轴选，　MPG_I2
13	Zsel	Z 轴选，　MPG_I3
6	Asel	第四轴轴选，　MPG_I4
14	Multi×1	倍率×1，，　MPG_I5
7	Multi×10	倍率×10，　MPG_I6
15	Multi×100	倍率×100，　MPG_I7
8	GND	

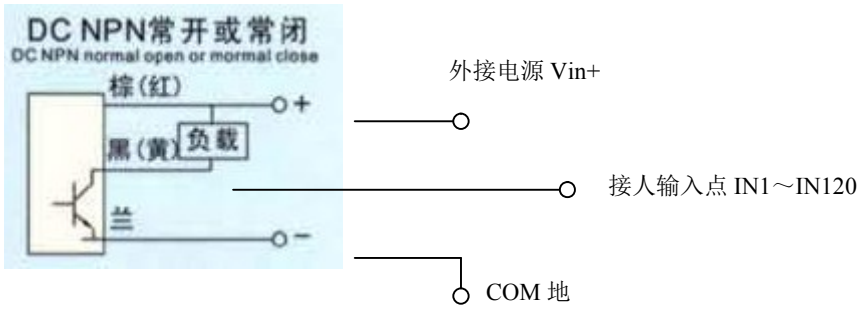
§1.8 通用数字 IO 信号

该部分输入点原理图适用于通用 IO 及 **附加编码器接口** 中的输入点。

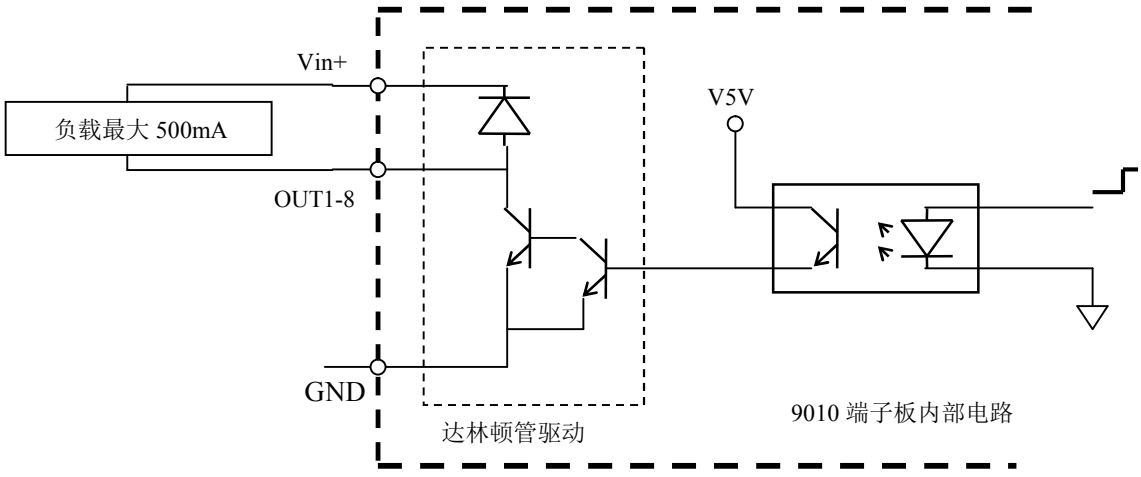
输入点原理图



当输入点是 光电开关类型的传感器时，请选用 NPN 型。 接线原理图如下：



输出点原理图



§1.9　防干扰措施

请务必做好以下防干扰措施！

① 伺服驱动器信号线缆均选用屏蔽线缆，并将伺服驱动器外壳与信号线缆屏蔽层有效连接。
② 9010 卡 15 针接头（轴控信号的接头）的外壳与伺服驱动器信号线缆屏蔽层有效连接。
③ 伺服驱动器外壳与 24V 直流电源共地。

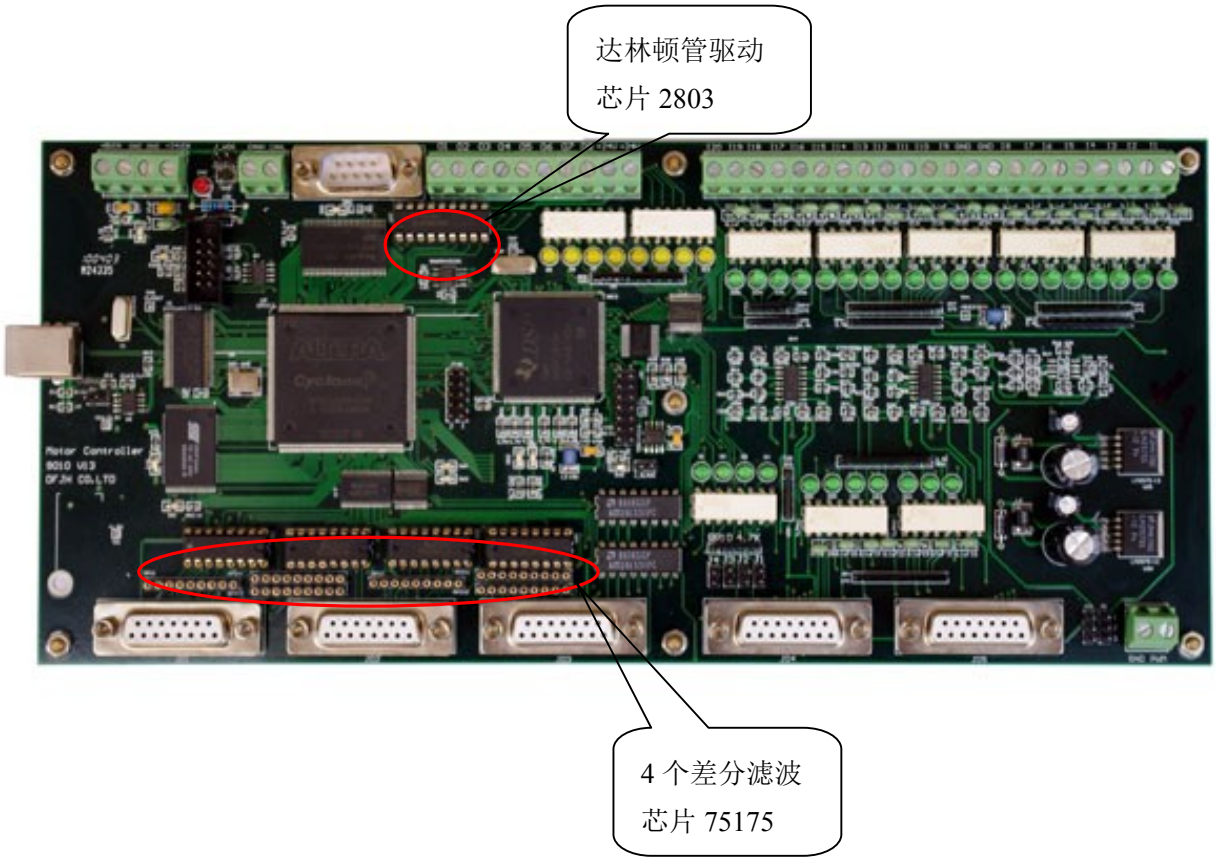
§1.10　9010 卡易损元器件

① 达林顿管驱动芯片 2803

正确连接 IO 输出可保证达林顿管驱动芯片 2803 的正常工作。

② 差分滤波芯片 75175

务必做好防干扰措施，否则可能因信号干扰造成差分滤波芯片 75175 的损坏。



第二章 编程指南

§2.1 一般说明

在随 9010 卡提供给用户的光盘中可找到以下文件夹：

[光盘]：------9010

|

|-----Driver

|-----Setup

|-----WinCNC

在 Driver 中提供了 9010 的驱动程序，包括 Win9x、Win2000、WinXp 三种平台。

在 Setup 中提供了 Windows 32 位 动态连接库 DLLs，支持 Microsoft Visual BASIC 、Visual C++及 Borland Delphi、C++ Builder。还有一些测试及示例程序。

在 WinCNC 中提供了基于 9010 卡的 CNC 机床数控软件 WinCNC for 9010 Demo（演示版）。

§2.2 软件的安装

在光盘的 9010\Setup 文件夹中运行 setup 程序即开始安装软件。setup 程序将提示安装目标盘和安装目录，该程序的省却安装目录为：

C:\9010

在该目录下，分别有 9010\VB、9010\VC、9010\Delph、9010\Bcb 子目录，它们分别对应 Microsoft Visual BASIC、Microsoft Visual C++、Borland Delphi、Borland C++ Builder 等编程环境。在这些目录下有以下文件：

表 2.1 9010\VB 文件一览表

文件名	解释
Declare.bas	VB 窗体函数声明文件
9010Test.EXE	9010 卡测试软件
9010Test	9010 卡测试软件的 VB 源程序

表 2.2 9010\VC 文件一览表

文件名	解释

dfjzh9010dll.dll	VC 动态联接库
dfjzh9010dll.lib	VC 联接库
dfjzh9010dll.h	VC 函数声明文件
9010Demo.EXE	9010 卡演示软件
9010Demo	9010 卡演示软件源程序

表 2.3　9010\Delphi 文件一览表

文件名	解释
Dfjzh9010Api.DLL	Delphi 动态联接库
drv9010.pas	Delphi 函数声明文件

表 2.4　9010\Bcb 文件一览表

文件名	解释
dfjzh9010dll.dll	C++ Builder 动态联接库
dfjzh9010dll.lib	C++ Builder 联接库
dfjzh9010dll.h	C++ Builder 函数声明文件
dfjzh9010dyn.h	C++ Builder 函数声明文件，动态调用 Dll 的头文件
Example	C++ Builder 例子文件夹

§2.3 编译与联接（Compiling and Linking）

这一节将说明如何使用 9010 的函数库，该函数库是以 Windows 32 位动态连接库 DLLs 的形式提供给用户的。

§2.3.1　Microsoft　Visual C++

用户在用 VC 编写程序时，首先将 9010 动态联接库 dfjzh9010dll.dll 拷贝到 Windows 系统目录，比如 C:\ Windows\System，或拷贝到 VC 的工作目录。然后将 dfjzh9010dll.lib 及头文件 dfjzh9010dll.h 包含在工程项目中，这样用户就可以使用 9010 的库函数了。

§2.3.2　Microsoft　Visual BASIC

用户在用 VB 编写程序时，首先将 9010 动态联接库 Dfjzh9010Api.dll 拷贝到 Windows 系统目录，比如 C:\ Windows\System，或拷贝到 VB 的工作目录。然后在窗体的通用说明区对 9010 库函数进行说明，这样用户就可调用 9010 的库函数对 9010 卡进行编程。

§2.3.3 Borland Delphi

用户在用 Borland Delphi 编写程序时，可按照如下步骤操作：

1）把 dfjzh9010Api.dll 拷贝到 Windows 系统目录，比如 C:\ Windows\System，或拷贝到当前工作目录；

2）把 drv9010.pas 文件拷贝到工作目录；

3）打开 delphi 编辑环境，打开应用程序项目文件；

4）打开 delphi 编辑环境中的“Project”菜单，点击“Add to project...”，把 drv9010.pas 文件添加到应用程序的项目文件中；

5）调出应用程序项目文件中使用 9010 卡工作的单元，点击“File”菜单，用该菜单下的“use unit ...”把 drv9010.pas 单元包括进去；

6）然后就像使用一般的系统库函数一样使用 dfjzh9010Api.dll 中的函数即可。

§2.3.4 Borland C++ Builder

在我们提供的 C++ Builder 应用 9010 的例子中，采用动态调用 Dll 的方法，C++ Builder 动态调用 Dll 的步骤如下：

1）从新定义函数。 我们已做，用户只需包含 dfjzh9010dyn.h 头文件即可。

2）载入 Dll，用 LoadLibrary 函数。

3）获取函数地址，用 GetProcAssress 函数。

4）强制类型转换，即将所获取的函数地址强制转换为函数。

5）函数调用。

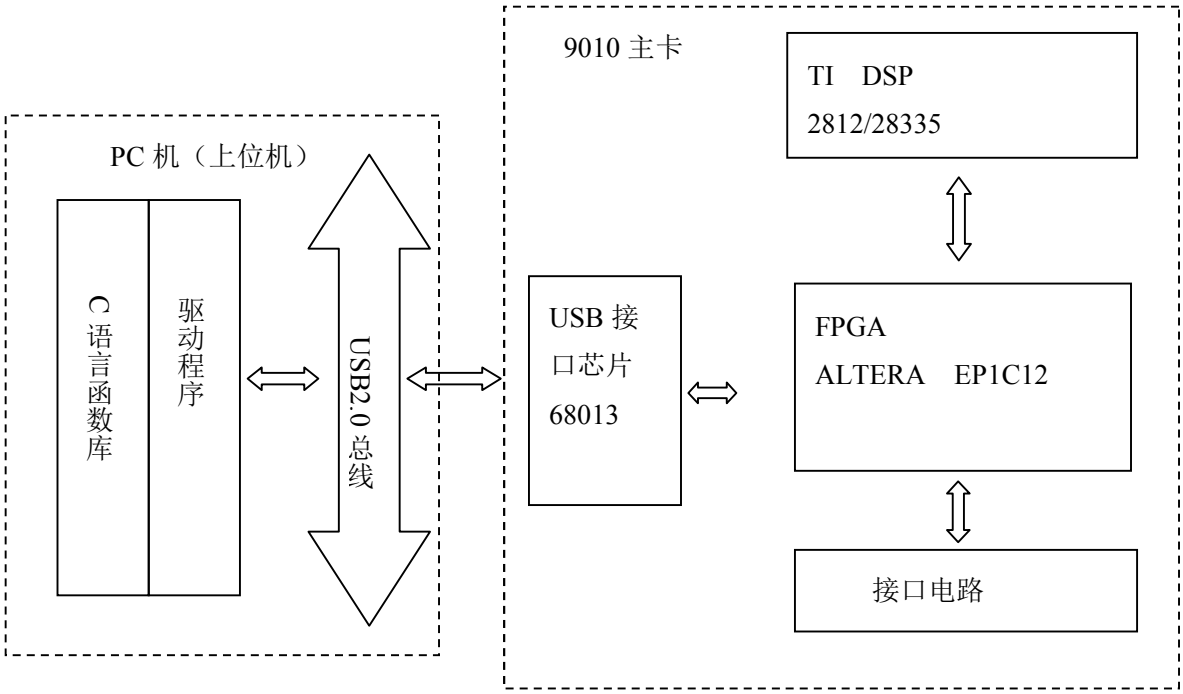
6）释放 Dll。

C++ Builder 动态调用 Dll 稍显复杂，更详细说明请看 C++ Builder 参考书（比如：《C++ Builder 编程技巧、经验与实例》人民邮电出版社）。

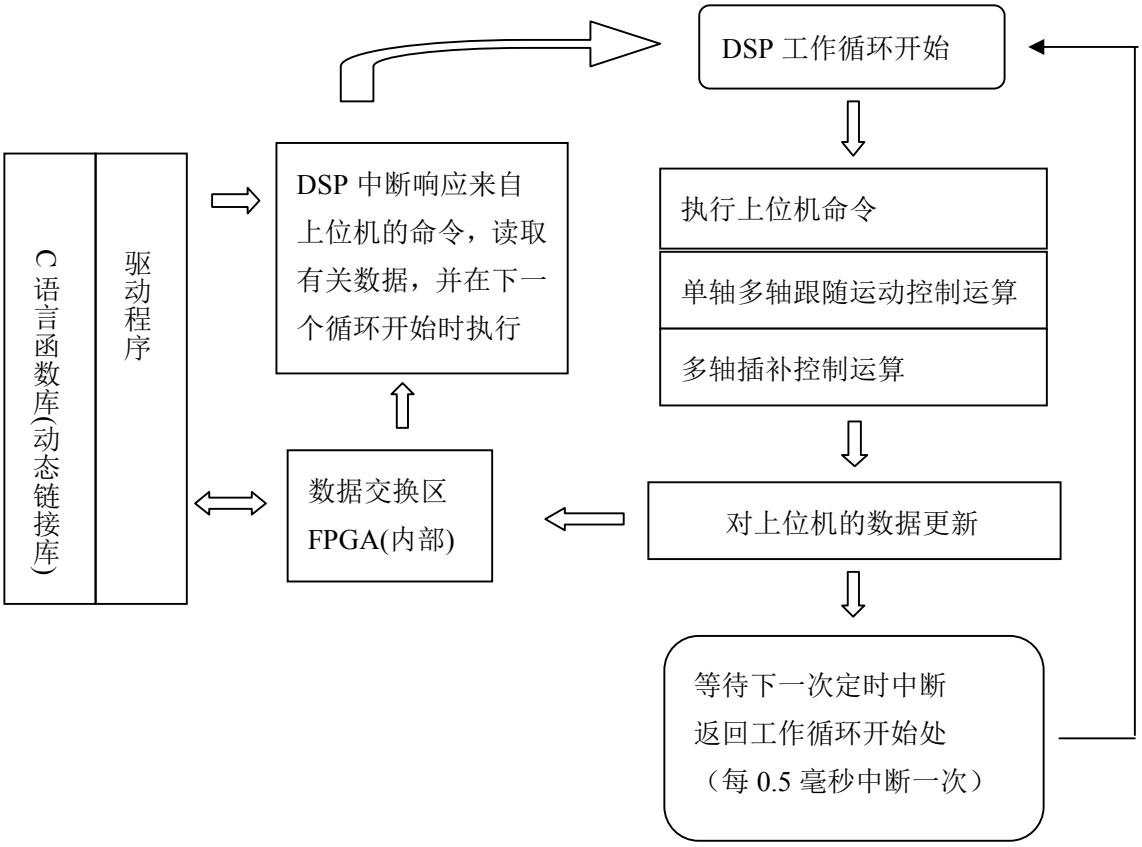
C++ Builder 静态调用 Dll 的步骤与 VC 类似，在此不再赘述。

§2.4 9010 卡工作原理

⑴9010 卡硬件工作原理框图



⑵9010 卡软件工作原理框图



如上图所示，在 FPGA 中开辟有 DSP 与 PC 机的数据交换区，PC 需要读取的数据（轴

位置、速度、轴状态、插补状态及 IO 等），DSP 会每 0.5 毫秒更新一次；而 PC 机在写入数据和命令时，先将数据写入数据交换区，然后向 DSP 发出中断信号通知 DSP 读取或执行命令。

§2.5 库函数介绍

9010 驱动函数库中总共有 100 多个函数，这些函数可粗分类为**初始化**函数、**单轴运动控制**函数、**多轴电子齿轮**函数、**检测**函数、**插补**函数、**NURBS 样条曲线插补**函数 及 **CAN 总线扩展**函数。每个函数的详细介绍在本书的第三章中给出。

在用户编制自己的程序时，我们建议用户参考我们所提供的示例程序，这些示例程序将使你省去很多例行工作。

§2.5.1 初始化函数

在调用函数库的各个函数之前，第一步必须对函数库进行初始化。初始化函数为 InitCrad_9010，该函数的第一个参数为板号，其它参数见第三章中详细说明。用户如果用到多个 9010 板，则每个板都必须初始化。 9010 的驱动程序最多支持 16 个轴，用户最多可同时使用 4 块 9010 板，板号为：0-3（保留功能）。

当在一台计算机中用到多块 9010 卡时，9010 卡的板号由其 USB 插槽位置决定（保留功能）。

在用户程序退出前，必须调用 ExitCard_9010()函数，释放函数库。当用到多个板时要一一释放，其释放次序是：最先初始化的最后释放，最后初始化的最先释放。

§2.5.2 轴工作模式设定函数

9010 卡的每个轴均可工作在如下模式：

0=轴无效;
1=开环脉冲模式;
2=位置闭环脉冲输出模式;
3=位置闭环模拟量输出模式;
4=单纯电压输出模式;
5=单纯 PWM 脉冲输出模式;

设定函数为：

SetAxisWorkMode_9010

硬件跳线（参看 *§1.5 电机控制信号* 节）决定函数 SetAxisWorkMode_9010 可设定范围，硬件跳线状态可用函数 GetAxisMode_9010 获得，当硬件跳线状态为 1 时（跳线不跨接），轴工作模式只能设 0、1、2 模式；为 0 时（跳线跨接），轴工作模式只能设 0、3、4、5 模式。

1=开环脉冲模式； **可接步进电机、伺服电机（伺服电机工作在位置模式下）。**

2=位置闭环脉冲输出模式； **可接步进电机（带编码器反馈元件的特殊步进电机）、伺服电机（伺服电机工作在位置模式下）。**

3=位置闭环模拟量输出模式； **可接伺服电机（伺服电机工作在速度模式下）。**

当轴工作在 1、2 模式下，轴的位置单位是脉冲数、速度单位是赫兹，这比较好理解，当轴模式 3 下，轴的位置单位英文名称是 count，中文姑且叫“当量”，其真实含义与伺服电机的编码器反馈线数（四倍频后）对应起来，比如伺服电机编码器反馈线数为 2500 线，设电机位置为 10000，则意味着伺服电机转一圈，所以“当量”也可以等价于脉冲数，速度也可以用赫兹表示。

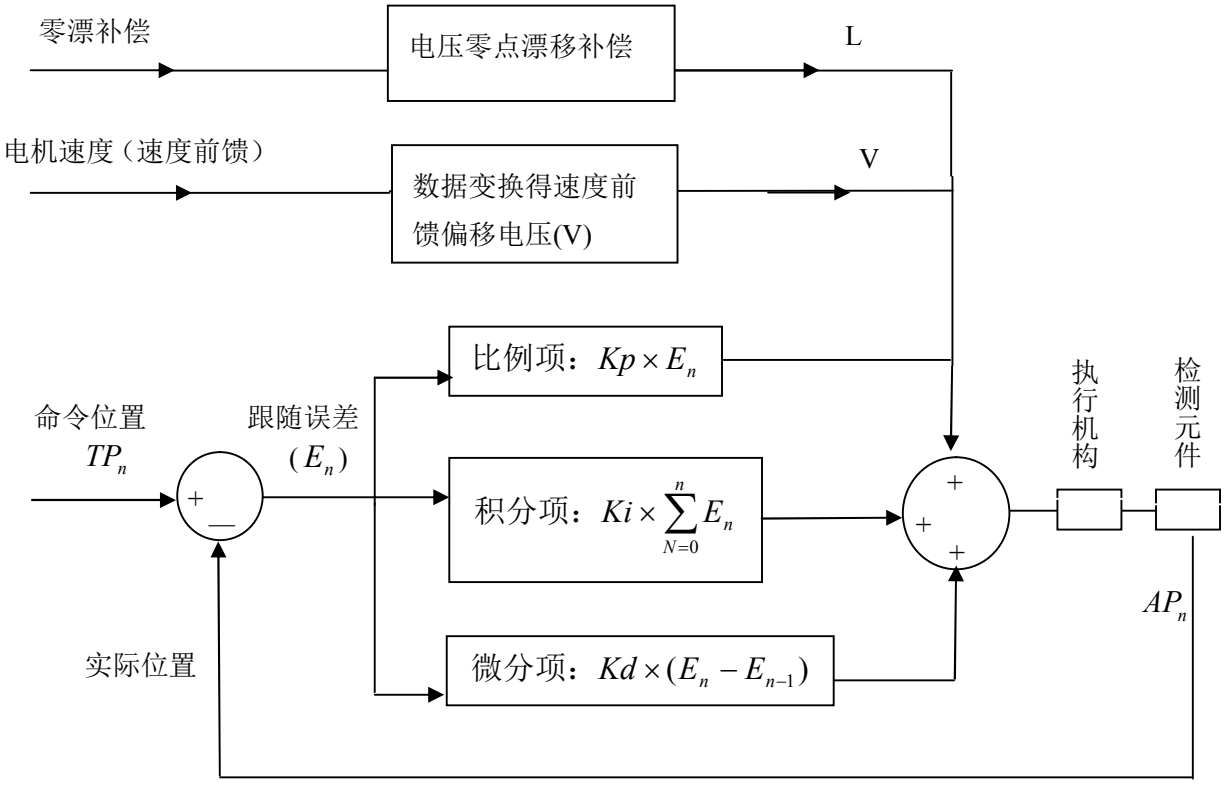
在本手册以下叙述中，无论轴工作在 1、2、3 那种模式下， 位置单位用脉冲数表示、速度单位用赫兹表示。

§2.5.3 轴位置闭环调节函数

当轴的工作模式为 3 时（位置闭环模拟量输出模式），要正确设定轴位置闭环调节各参数值。

1、 闭环控制理论

9010 卡用智能 PID 数字调节器组成伺服电机的闭环控制系统，控制框图如下：



跟随误差 $= PositionError_n = E_n = TP_n - AP_n$

输出 $= Output_n = (Kp \times E_n) + Ki\sum_{N=0}^{n} E_n + Kd(E_n - E_{n-1}) + L + V$

其中：

n 是采样时间

TP_n是在采样时间n时的命令位置

AP_n是在采样时间n时的实际位置

从以上公式中可以看到：

KP 值（比例项）：**对当前的跟随误差比较敏感。与时间无关**。即时成比例地反映控制系统的跟随误差，误差一旦产生，调节器立即产生控制作用以减小偏差。

KI 值（积分项）：**对跟随误差的累加值比较敏感。与时间有关，与轴的控制历史有关**。主要用于消除静态误差，提高系统的无差度。

KD 值（微分项）：**对最近两次的跟随误差的差值比较敏感。与时间有关**。能反映跟随误差的变化趋势（变化速率），并能在跟随误差的值变得太大之前，在系统中引入一个有效的早期修正信号，从而加快系统的动作速度，减小调节时间。

2、有关参数设定函数

零漂补偿值设定函数：

SetAxisOPC_9010

PID 参数设定函数：

SetAxisKP_9010
SetAxisKI_9010
SetAxisKD_9010
SetAxisIL_9010

速度前馈参数设定函数：

SetAxisFVRate_9010
SetAxisFV_9010

闭环控制参数调节可分为三步完成：

第一步：设定“零漂补偿” L 值：

调节范围一般在 －40～40；当调节到轴位置的理论值与实际值差在较小范围时，可转到下一步。

第二步：设定“PID 调节”各参数值：

KP值：通过调节KP值，将轴位置的理论值与实际值差在较小范围即可。
设置 KP 值时，可先将 KP 值设为 1，然后根据轴的运动品质逐步增加 KP 值。不可突然加大 KP 值，否则轴有可能产生剧烈震荡。
KI值：一般设为0。
KD值：一般设为0。

第三步：设定“速度前馈”V 值

根据伺服电机实际参数，通过函数 SetAxisFVRate_9010 正确设定伺服电机**4 倍频前编码器线数**及 **命令电压对应轴电机转速** 值；然后通过函数 SetAxisFV_9010 可设定“速度前馈”调节的百分率，可设置为100（即100%速度前馈）。

3、 如何得到“轴位置的理论值与实际值误差”

9010 卡初始化后，假如第 0 号轴工作在“位置闭环模拟量输出模式”下，如下 C 语言片断：

```
long    Pos1,Pos2;
Pos1= ReadAxisTheoryPos_9010 (0,0);       //读取轴理论位置值
Pos2= ReadAxisEncodePos_9010 (0,0);       //读取轴编码器反馈值（实际值）
```

Pos1 与 Pos2 的差值就是：轴位置的理论值与实际值差，也就是轴的 “***跟随误差***”。函数：

SetAxisPEL_9010

用来设定***跟随误差*** 的最大极限值，当轴的***跟随误差*** 超过最大极限值时，9010 卡将自动将该轴控制电压输出为理论 0 伏（0 +零漂补偿值），并产生报警号，函数 GetErrorNo_9010 将返回 36（轴位置跟随误差超限），函数 ReadAxisState_9010 将返回-13 值。

注意事项：

1、函数 SetAxisFVRate_9010设定伺服电机的参数时，由于伺服电机有“**电机最高转速**”与“电机额定转速”。

在此处闭环参数设置中请务必正确填入“**电机最高转速**”(9010 卡轴控信号输出10V 时,对应的伺服电机速度)。

在确认正确填入伺服电机**4 倍频前编码器线数**及**电机最高转速**后，FV 值可设为 100（意为 100%速度前馈，这样伺服电机的动态误差最小）。如果 **4 倍频前编码器线数**或**电机最高转速**都不能确定正确与否，FV 值只能从 1 到 100 测试，以电机能平稳运行为标准。

2、IL(积分极限)参数：

IL 值（积分极限）对积分项 $\sum_{N=0}^{n} E_n$ 设最大限制。

IL 值（积分极限）一般可设为 200-500。

积分项大于该设定值时，积分项的计算输出值将为 0，其目的是为了减小系统过调。

3、 PID

当用户将某一轴的 KP、KI、KD 全设为 0 时，该伺服轴将组成不了位置闭环控制，该轴将不完全受电机控制卡的控制（只受速度前馈控制，如果不为 0）。

通过调节 KI 值可减小轴的静态误差，一般设置为 0。特殊情况时，用户可通过调节 KI 值减小轴在静止状态下的定位误差。

KD 一般可设为 0，适当增加 KD 值可改善轴加速时的动态响应品质，一般原则是 KD 值不要大于 KP 值。

切记：要<u>逐步增加</u> KP、KI、KD 的各项值，不可突然加大 KP、KI、KD 的值，否则轴有可能产生剧烈震荡。

4、 编程顺序

对于闭环控制轴，编程顺序的一般原则是：

初始化 9010 卡：

调用 InitCard_9010 (….)函数

⇩

设轴编码器倍频数：（如果需要 4 倍频，可忽略本步骤）

调用 SetEncoderX4X1_9010 (….)函数

设 **零漂补偿、PID、速度前馈、*跟随误差限*** 等：
通过如下函数：

SetAxisOPC_9010 (….);
SetAxisKP_9010
SetAxisKI_9010
SetAxisKD_9010
SetAxisIL_9010
SetAxisFVRate_9010
SetAxisFV_9010
SetAxisPEL_9010

设轴工作模式为 3：
调用 SetAxisWorkMode_9010 (….)函数
（这时 **轴闭环调节控制系统** 建立）

继续…
往后也可随时改变 ***零漂补偿、PID、速度前馈、跟随误差限*** 等值，
注意轴是否在停止状态。以后禁止改变轴编码器倍频数。

§2.5.4 单轴运动控制函数

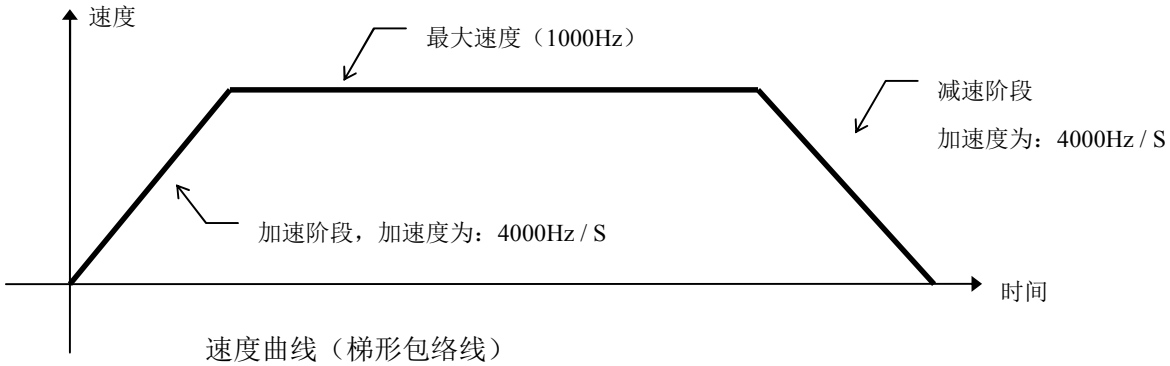
9010 板提供了轴的两种基本运动形式，即轴的点到点的运动控制（**位置模式**）和速度控制（**速度模式**）。

位置模式时，当用户在程序中写下如下语句：

SetAxisAcc_9010(0,0,4000);

```
SetAxisVel_9010(0,0,1000);
SetAxisPos_9010(0,0,xPosSet);
StartAxis_9010(0,0);
```

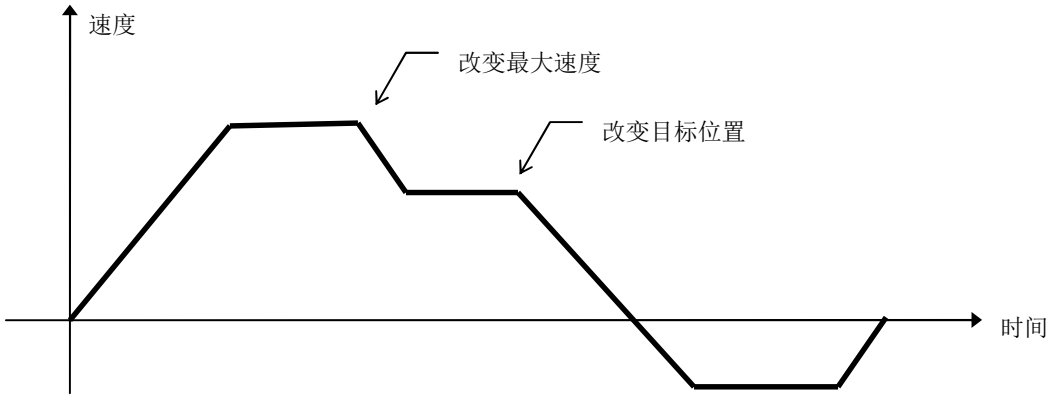
0 号轴将以最大 1000Hz 的频率发 2000 个脉冲，速度曲线如下：



速度曲线（梯形包络线）

上图为梯形包络线，它的加减速值相等。该梯形的面积就是电机转动的距离（2000 个步距角）。由于步进电机所带负载有一定的机械惯性质量，速度不可能立刻达到设定值，所以应有加减速阶段。

在电机运行时，可随时改变梯形包络线的参数，可走出更复杂的运动轨迹，如下图所示：

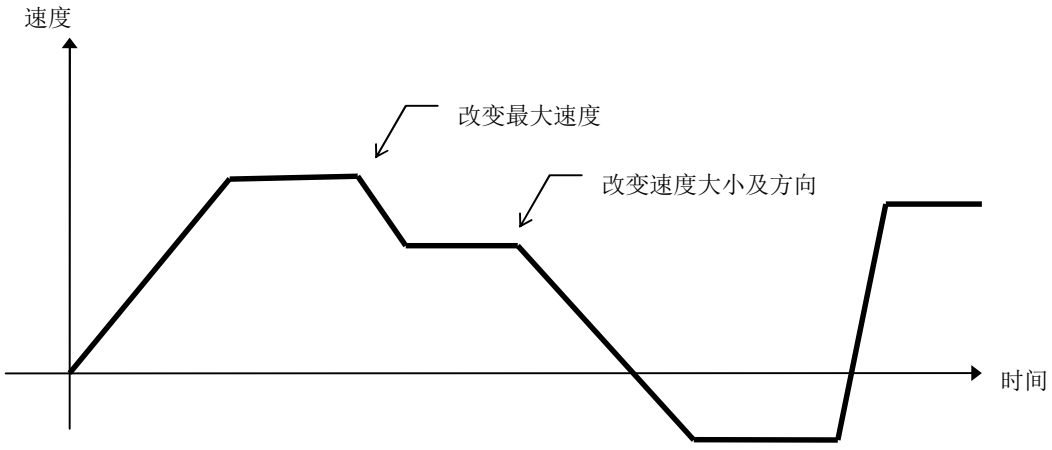


复杂的梯形包络线

用户也可以用速度模式来控制电机轴运动，速度模式函数为：

short StartAxisVel_9010 (unsigned short Board_NO,unsigned short Axis_No,long velocity);

该函数的第一个参数为卡号 0、1、2……，第二个参数为轴号 0、1、2……，第三个参数为速度参数，单位为脉冲频率（Hz 脉冲数/秒），可设正负。速度模式下的电机速度曲线如下：



速度模式曲线

速度模式的曲线类似于位置模式，但它没有终点值，用户可随时用轴停止命令函数（AbortAxis_9010、StopAxis_9010、CeaseAxis_9010）命令电机停止运行。在实际编程当中，用户可随时改变步进电机为位置模式或为速度模式。

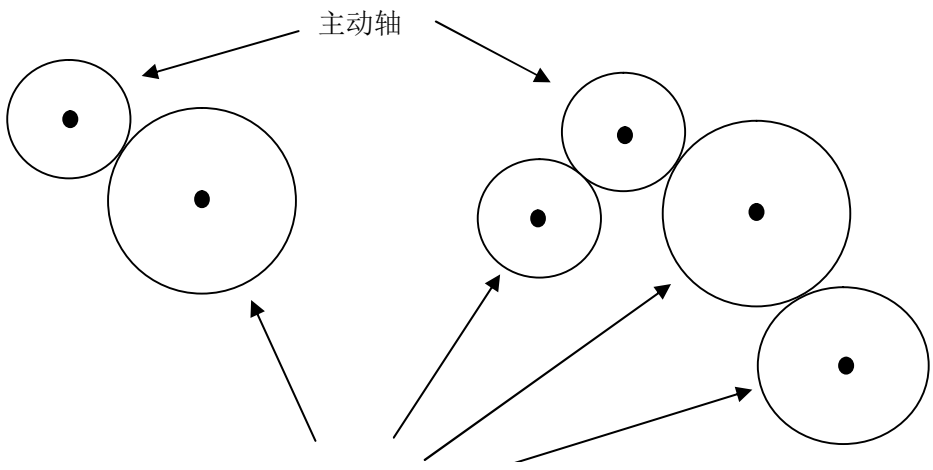
§2.5.5 轴随动运动控制函数

9010 板提供了轴的两种两种随动运动形式，即从动轴随动主动轴（**电子齿轮**）、轴随动编码器（**编码器控制模式**）。

在电子齿轮模式下，通过函数：

SetAxisEGear_9010(Board_NO, Axis_No,F_Axis_No,Rate,Kp)；

来设定主从关系、齿轮比等。主从关系可随意设定，只要不违反自己随自己或父子孙的伦理关系就可以。齿轮比很广，在 0.001 至 1000 之间，可正负。电子齿轮随动关系，举例如下图：



在编码器控制模式下，通过函数：

SetAxisFE_9010(Board_NO, Axis_No,Rate,Kp, Mode1, Mode2,Mode3)；

来设定轴跟随编码器，可以是速度方式或位置方式等。另外，在插补模式下也可设轴跟随编码器，这为用户编写带螺纹切削功能的数控系统提供方便。

§2.5.6 位置捕捉函数

9010 控制卡提供了 4 个轴的位置捕捉功能。具体函数和使用步骤如下：

1. 通过函数 SetAxisIO_9010 设置 轴位置捕捉 触发信号的输入点。
2. 通过函数 IsPosCatch_9010 检测轴位置捕获标志位。
3. 通过函数 ReadCatchPos_9010 读取触发时刻的轴位置。
4. 通过函数 ResetPosCatch_9010 复位轴位置捕获标志位，准备下一次位置捕捉。

注意事项：

（1）每个轴的 **位置捕捉** 触发信号输入点的位置固定不变：输入点 I17、I18、I19、I20（参看 ***§1.4 控制卡的硬件连线*** 节）分别对应 0、1、2、3 轴。

（2）由于硬件资源的关系，9010 的每个轴实际只有两个 32 位 **位置寄存器**，它们分别是 轴 ***理论位置值位置寄存器***（函数 ReadAxisTheoryPos_9010 读取）， 轴***编码器反馈位置寄存器***（函数 ReadAxisEncodePos_9010 读取）。当轴位置捕获标志位有效时（通过函数 IsPosCatch_9010 获得），函数 ReadCatchPos_9010 将会返回触发时刻的轴位置，这时轴的捕捉位置将临时占据轴实际两个 **位置寄存器** 中的的一个。如前所述，9010 卡的每个轴均可工作在如下模式：

0=轴无效
1=开环脉冲模式
2=位置闭环脉冲输出模式
3=位置闭环模拟量输出模式
4=单纯电压输出模式
5=单纯 PWM 脉冲输出模式

在模式 1 时，捕捉位置值 将临时占据轴的 ***编码器反馈位置寄存器***，在模式 2、3、4、5 时，捕捉位置值 将临时占据轴的 ***理论位置寄存器*** ，函数 ResetPosCatch_9010 将解除这种临时占据。

§2.5.7 插补函数

在实际应用中，有可能需要几个电机轴按一定轨迹同时运动，最典型的是数控机床应用，比如：在数控铣床中的 XYZ 坐标系中，要求机床的 XYZ 轴运行一段直线或一段圆弧或一段空间螺旋线等，这种 ***几个电机轴按一定轨迹同时运动*** 的方式就是所谓的插补运动。

在 9010 卡的插补引擎中虚拟了四个轴，即 X、Y、Z、W 轴，组成一个 XYZW 坐标系。这四个虚拟轴如何对应卡上的四个实际轴完全由用户控制，而且它们的编程单位是按实际应用单位实现的（即：用户单位，由用户指定是 毫米、 还是角度或是角弧度等）。针对数控机床的实际应用，9010 卡对插补函数进行了如下优化：

1. 虚拟坐标轴，用户可方便组成 2 轴、3 轴或 4 轴数控系统
2. 用户单位编程，编程直观，中间单位转换的计算量最小。
3. 绝对值编程，没有累计误差和计算误差
4. 小线段连续插补算法先进，在曲线或曲面加工时有显著优势。
5. 实现不同平面的圆弧插补和螺旋线插补。
6. 实现 4 轴线性和 4 轴圆弧（2 轴直线；既螺旋线）插补。
7. 实现插补模式下的轴位置跟随编码器（螺纹切削、硬攻丝）。
8. 实现 3 轴、4 轴 NURBS 曲线插补插补。

9010 插补函数库相对独立，所有插补函数均以字母 LM 打头，按作用可分为**参数设定**函数、**状态检测**函数、**实际操作**函数、**插补顺序执行**函数。它们的函数原型如下：

参数设定函数：

设置插补引擎参数：short LM_SetMinVel_9010 (….);
设置插补引擎参数：short LM_SetMaxVel_9010 (….);
设置插补引擎参数：short LM_SetSpeedPri_9010 (….);
设置插补引擎参数：short LM_SetParaAngle_9010 (….);
设置插补引擎参数：short LM_SetSysPara_9010 (….);
设置插补引擎参数：short LM_SetAxisMaxErrLtd_9010(….);

设置某轴与插补 X 匹配： short LM_SetXAxis_9010 (….);
撤消插补 X 匹配： short LM_OffXAxis_9010 ();
设置某轴与插补 Y 匹配： short LM_SetYAxis_9010 (….);
撤消插补 Y 匹配： short LM_OffYAxis_9010 ();
设置某轴与插补 Z 匹配： short LM_SetZAxis_9010 (….);
撤消插补 Z 匹配： short LM_OffZAxis_9010 ();

设置某轴与插补 W 匹配： short LM_SetWAxis_9010 (….);
撤消插补 W 匹配： short LM_OffWAxis_9010 ();

设置插补加减速： short LM_SetACCDec_9010 (….);
设置插补 S 形加减速： short LM_SetSAccePower_9010 (….);

设置 Nurbs 曲线插补预处理的扫描模式：
short LM_SetNurbsScanMode_9010(….);
设置 Nurbs 曲线插补时的速度控制：
short LM_SetNurbsVelCtrl_9010(….);
设置 Nurbs 曲线插补加速度和减速度：
short LM_SetNurbsAccDec_9010(….);
设 Nurbs 曲线插补误差补偿系数：
short LM_SetNurbsCompCoef_9010(….);

状态检测函数：

返回插补命令缓冲区长度： short LM_GetBuffLen_9010 (….);
返回插补状态： short LM_GetState_9010 (….);
返回插补测量状态： short LM_GetMeasureState_9010 (….);
返回当前插补行号： short LM_GetLineNO_9010 (….);
返回插补编码器跟随状态： short LM_GetFEnState_9010 (….);

读取 NURBS 曲线插补运行参数值:
float LM_GetNubrsExecPara_9010(….);
获得在插补缓冲区中有几条 Nurbs 曲线：
short LM_GetNurbsInBuffLen_9010(….);
获得产生 Nurbs 曲线插补错误的编号:
short LM_GetNurbsErrorNo_9010(….);

实际操作函数：

清空插补缓冲区： short LM_CleanBuff_9010 (….);
暂停插补运算： short LM_Pause_9010 (….);
恢复插补运算： short LM_Resume_9010 (….);
变更插补速度： short LM_SetSpeedRate_9010 (….);
插补到当前行结束停止： short LM_LineEnd_9010(….);
插补开始： short LM_Start_9010(….);

获得插补开始位置

X 轴： short LM_GetXStartPos_9010(….);

Y 轴： short LM_GetYStartPos_9010(….);

Z 轴： short LM_GetZStartPos_9010(….);

W 轴： short LM_GetWStartPos_9010(….);

向 9010 卡传输 NURBS 曲线数据:

short LM_SendNurbsData_9010(….);

插补顺序执行函数：

插补结束命令： short LM_End_9010 (….);

线性插补命令： short LM_Line_9010 (….);

线性插补命令： short LM_LineMaxV_9010 (….);

线性插补测量命令： short LM_LineMeasure_9010 (….);

线性插补跟随编码器位置： short LM_LineFE_9010 (….);

圆弧/螺旋线插补命令(顺圆)：

XY 平面：short LM_ArcCW_9010 (….);

ZX 平面：short LM_ArcCW_ZX_9010 (….);

YZ 平面：short LM_ArcCW_YZ_9010 (….);

圆弧/螺旋线插补命令(逆圆)：

XY 平面：short LM_ArcCCW_9010 (….);

ZX 平面：short LM_ArcCCW_ZX_9010 (….);

YZ 平面：short LM_ArcCCW_YZ_9010 (….);

NURBS 曲线插补命令：

初始化 Nurbs 曲线： short LM_NurbsInit_9010 (….);

设 Nurbs 曲线插补数据： short LM_NurbsData_9010(….);

Nurbs 曲线插补命令（3 轴）： short LM_Nurbs_9010(….);

Nurbs 曲线插补命令（4 轴）： short LM_Nurbs4Axis_9010(….);

插补等待命令： short LM_Wait_9010 (….);

插补等待输入点命令： short LM_Wait_I_9010 (….);

插补输出 IO 点命令： short LM_IOOut_9010 (….);

插补输出 PWM 命令： short LM_PWM_9010 (….);

有关 NURBS 曲线计算的函数：

初始化 Nurbs 曲线,为 Nurbs 曲线计算做准备:

short Set_NurbsInit_9010(….);

设 Nurbs 曲线计算数据: short Set_NurbsData_9010(….);

设 Nurbs 曲线数据结束: short Set_NurbsEnd_9010(….);

计算 Nurbs 曲线轴位置(型值点)坐标:

short Get_NurbsPos_9010(….);

计算 Nurbs 曲线长度: double Get_NurbsLen_9010(….);

获得产生 Nurbs 曲线计算错误的编号:

short Get_NurbsErrorNo_9010(….);

使用插补函数的一般步骤如下：

1. 设置插补引擎参数、设置某些轴与插补虚拟 XYZW 轴匹配、设置插补加速度和减速度。
2. 获得插补轴开始位置（通过 LM_GetXStartPos_9010 等命令）
3. 写入插补顺序执行命令（通过 直线、圆弧等插补命令 写入插补缓冲区）。
4. 插补开始（通过 插补开始命令 LM_Start_9010（….）;
5. 根据需要，循环或定时检测插补缓冲区，如果不满，可继续写入插补顺序执行命令。

注意：在步骤 1 和 2 时，注意相关轴必须是停止状态。

在写入插补顺序执行命令时，是写入插补缓冲区中，该插补缓冲区的长度为 256 行，即一次最多可写 265 行插补顺序执行命令。插补引擎会按照写入顺序一行一行执行，直到碰到插补结束命令（LM_End_9010）或插补缓冲区空。用户也可用轴停止命令（AbortAxis_9010、StopAxis_9010、CeaseAxis_9010）来终止插补引擎工作。在插补引擎工作时也可随时暂停插补轴运动（用 LM_Pause_9010 命令）和恢复插补运动（用 LM_Resume_9010 命令）。在插补轴运动时也可用 LM_SetSpeedRate_9010 命令随时改变插补速度。

§2.5.8 插补疑问

（1）插补运算的单位是什么

距离是：用户单位；速度是：用户单位/分钟；加速度是：用户单位/秒平方

在用户设置某轴与插补虚拟的 X、Y、Z、W 轴进行匹配时（用 LM_SetXAxis_9010 及 YZW 各轴相关函数 ），函数 LM_SetXAxis_9010 中将代入该轴的脉冲当量，脉冲当量就是：该轴的电信号发一个脉冲，对应该轴实际移动多少用户单位。

比如，某一直线轴通过滚珠丝杠与电机直联，该滚珠丝杠的螺距是 5 毫米，而电机转一

圈需 2000 个脉冲，则：轴的脉冲当量=5/2000=0.0025；0.0025 的含义是：一个脉冲将使该轴移动 0.0025 毫米。

轴的脉冲当量很重要，轴的脉冲当量确定后，则该轴的用户单位就确定，在上例中轴的用户单位就是毫米。

在实际插补中，插补速度是几个轴的合成速度，即矢量速度。这种情况稍微复杂，解释如下：

比如，X、Y 两轴要插补运动，X、Y 联在一个十字滑台上，即它们都是直线运动轴，它们的脉冲当量有可能不一样，但它们的用户单位是一样的，都是毫米，这时它们的插补矢量单位也是毫米，毫不含糊。矢量速度就是：毫米/分钟。

如果，X、W 两轴要插补运动，X 联在一个直线轴，W 联在一个旋转轴上，这时不管它们的脉冲当量是否一样，但它们的用户单位是不一样的，一个是毫米，一个是度，这时它们的插补矢量单位就很不直观，是毫米？还是度？不管怎样理解，它们的插补运动按设置的轨迹严格执行的。

例如编程如下（C 语言）：

```
LM_SetXAxis_9010(0,0,0.0025,0.01);     //0 轴与 X 轴匹配，脉冲当量是：0.0025,直线轴
LM_SetWAxis_9010(0,3,0.089,0.01);      //3 轴与 W 轴匹配，脉冲当量是：0.089, 旋转轴
LM_SetACCDec_9010 (0,5000,5000);   //设置插补加速度和减速度
LM_GetXStartPos_9010(0,NULL);          //获得 X 轴插补开始位置，假如是 0
LM_GetWStartPos_9010(0,NULL);          //获得 W 轴插补开始位置，假如是 0
LM_Line_9010 (0,100,0,0,360,1000,100);    //X 轴终点位置是 100，W 轴是 360，
                                            //速度 1000/分钟
LM_End_9010(0);                            //插补结束
LM_Start_9010(0,0);                        //开始插补
```

上例中，虽然插补矢量单位不明确，但并不影响 X、W 两轴准确的按线性比例的运行到 100、360 的位置。

（2）在插补开始前，为什么要获得轴插补开始位置

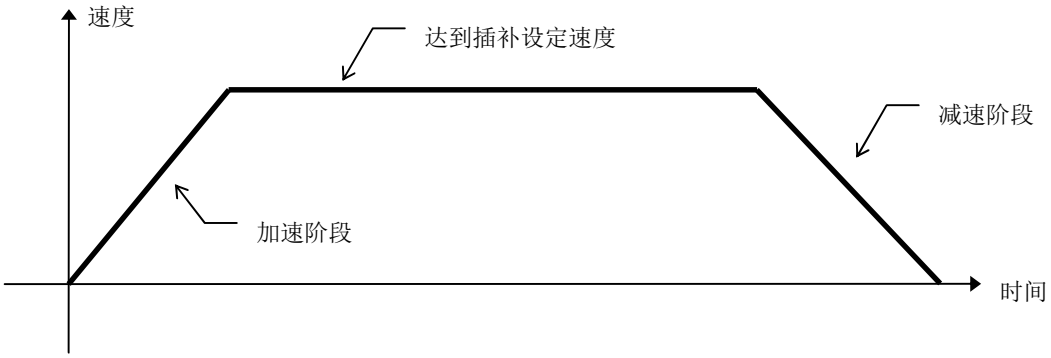
9010 卡的插补运算是在一个虚拟的 XYZW 坐标系中进行，其插补函数位置坐标全是绝对坐标，所以在插补开始前，首先要确定起点。函数 LM_GetXStartPos_9010（及对应的 Y、Z、W 轴函数）指示 9010 卡，X、Y、Z、W 轴的当前位置为插补起始点。在后面可以看到，插补顺序执行函数所代入的轴位置参数都是轴的终点位置值，所以在插补开始前，显式指示 9010 卡获得插补开始位置（即：告诉 9010 卡，轴的当前位置是插补开始位置）就是很自然的事情。

目前在市场上流行的运动控制卡，绝大部分用相对坐标来进行插补运算，但在实际运用

中（典型例子是 CNC 机床数控系统），绝大部分是用绝对坐标编程的，针对实际应用，9010 卡选择用绝对位置编程，且用 **用户单位** 编程。这样作有很多优点，它方便了用户编程，省去了很多中间计算环节，非常直观，且没有积累误差。

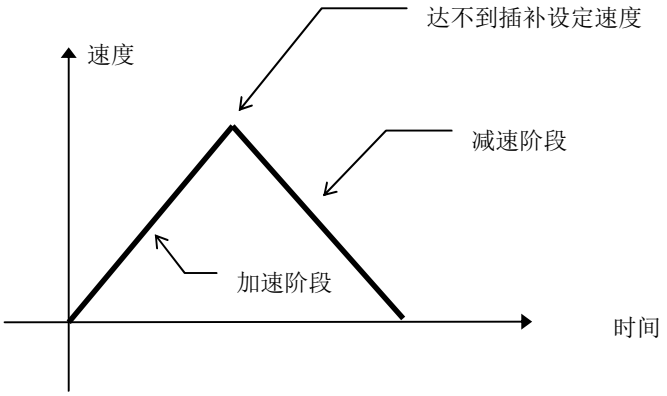
（3）9010 卡怎样处理插补加速度及插补速度

简单一条直线或圆弧，它们的加减速按直线处理，加速度和减速度可以不一样，如下：



插补速度曲线

如果直线或圆弧很短，不够加减速，则插补速度曲线如下：

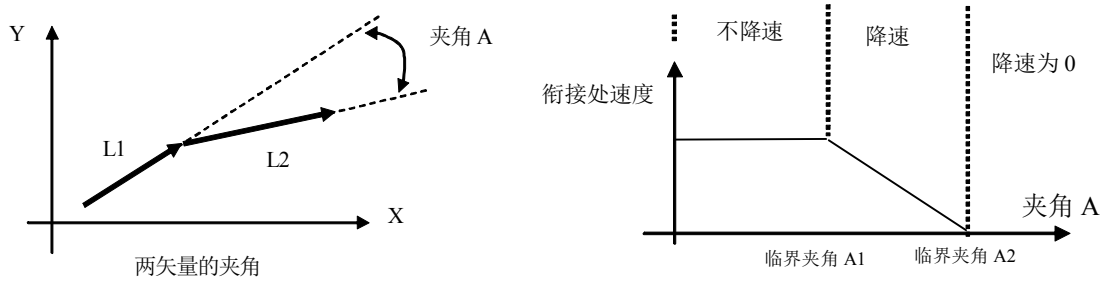


插补速度曲线

（4）两线段之间如何进行插补速度衔接

这里所谓“线段”是指 **直线** 或 **圆弧** 或 **Nurbs 曲线**。9010 卡在处理两线段之间速度衔接时，采用矢量夹角的方法进行处理，即根据两线段的矢量夹角来决定衔接处的插补速度。如下图所示：

设线段 L1 与线段 L2 的插补速度是一样的，均为 F，两线段的矢量夹角为 A，由函数



LM_SetParaAngle_9010（….）设定了两个临界夹角：A1 和 A2，则线段 L1 终点处的速度 Fe 计算如下：

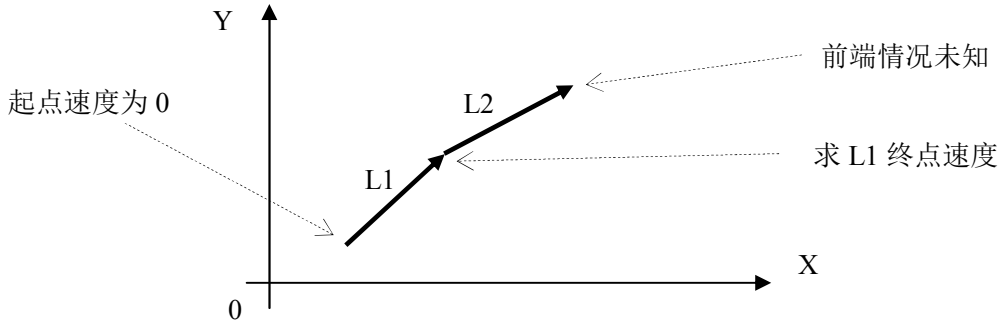
Fe=F；　当 A≤A1

Fe=0；　当 A≥A2

$$F_e = F \times \frac{A2 - A}{A2 - A1}$$　　当 A1≤A≤A2

（5）多个小线段连续插补时如何处理插补速度

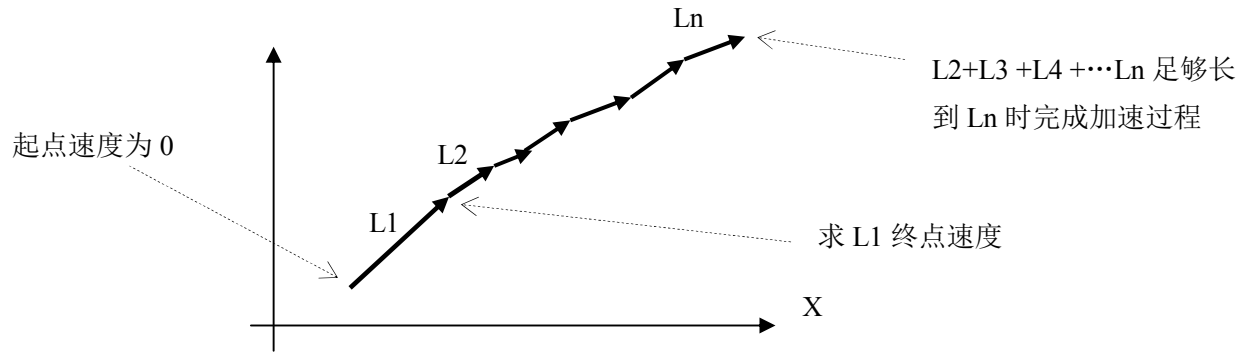
所谓小线段是指它的长度不足以按设定加减速完成加速过程或减速过程，加速过程是指从某一起始速度（比如 0 速）加速到指定速度，同理减速过程。通常插补引擎在运行到某一线段之前，要对它进行插补预处理，要预知它的起点速度和终点速度，线段首尾相连，本段的终点速度就是下一段的起点速度，所以在插补预处理中主要是计算线段的终点速度。当线段很短时，情况就变得复杂起来，举例如下：



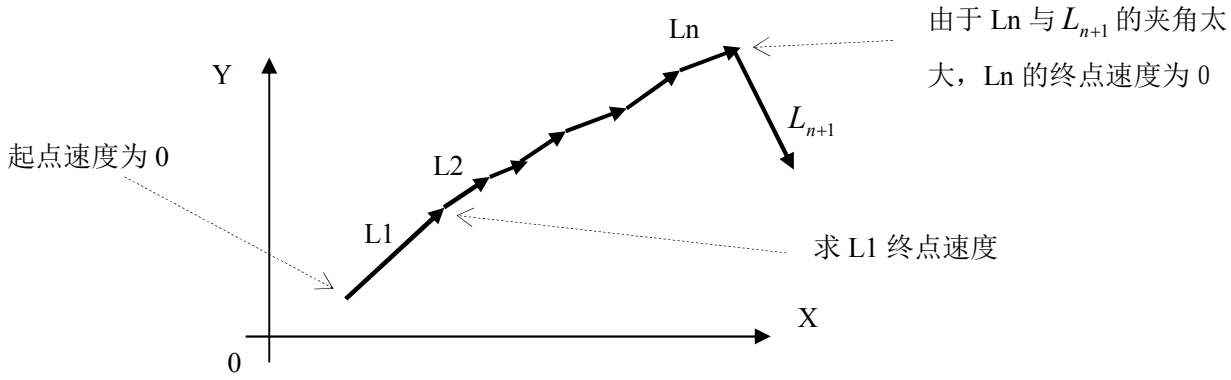
如上图，假如 L1 、L2 矢量夹角很小，L1 是短线段（不足以完成加速过程）。按合理的推断 L1 、L2 衔接处的速度（L1 终点速度）应该平滑过渡，假如 L2 也很短（不足以完成加速过程），L1 终点速度到底是多少就很难推算。要推算 L1 的终点速度，必须满足以下条件：

1．L2 足够长

2．L2+L3 +L4 +…Ln 足够长，如下图：



3．L2+L3 +L4 +…Ln 不足够长，但 Ln 的终点速度可知，如下图：



不难看出，满足以上 3 种条件，就能计算出 L 1 终点处的速度。

9010 卡的插补缓冲区有 256 行深度，当用户不断将插补数据写入缓冲区时，也就不断的激励 9010 卡进行插补预处理，由于 9010 卡采用以上所介绍的插补算法，使 9010 卡有能力进行多个小线段的连续高速插补。

在极端情况下，如果 256 行的小线段都不足以完成加减速过程，这时该如何处理呢？在 9010 卡的插补函数：LM_Start_9010(….)中有强制执行参数：ObligeFlag；当 ObligeFlag 设为 1 时，将假定第 265 行的（或最末端的）终点速度为 0，这样就能计算出各线段的终点速度，使插补引擎能够连续运行。

（特别声明：256 行的插补预处理深度（插补缓冲区），在绝大部分情况下已经足够了，根据用户需要我们可提供 512、1024、2048 或最高 4096 行插补预处理深度的个性版软件。）

（6）什么时候发出插补开始命令 LM_Start_9010

1．当插补缓冲区空时，用户然后写入插补顺序执行命令，这时发出插补开始命令 LM_Start_9010 将启动插补引擎。

2. 当插补引擎执行到 LM_End_9010 后，需要发出插补开始命令 LM_Start_9010 再次启动插补引擎。
3. 当发出 LM_LineEnd_9010 命令后，插补引擎运行到当前行结束并停止，需要发出插补开始命令 LM_Start_9010 再次启动插补引擎。
4. 当插补引擎被轴停止命令（StopAxis_9010、AbortAxis_9010、CeaseAxis_9010）终止后， 需要再次启动插补引擎，这时用户需要先清空插补缓冲区空，形成条件 1。

（7）在插补开始运行时，轴单独的控制命令如何运用

在相关轴进行插补运算时，除了轴停止命令（StopAxis_9010、AbortAxis_9010、CeaseAxis_9010），其它命令都不能执行。如果用户写入其它命令（比如 StartAxis_9010），系统将报错，用函数 GetErrorNo_9010 可以检测到错误号。

与插补不相关的轴（即：轴没有与插补虚拟轴 XYZW 进行匹配），可以正常操作。

（8）如何实现不同板间的的多轴插补（保留功能）

理论上不同板间的轴是不能实现插补的，因为插补运算是由板上的 DSP 完成的，举例说明，假如用户用到两块 9010 卡，要实现 8 个轴的线性插补，这时用户要自行计算两块卡上插补矢量的分量，并分别写入各自板上的插补缓冲区，而后顺序启动两板的插补引擎，这时从微观来看，两块板上的插补引擎启动时间不能同步，最多相差在 2 毫秒之内（由 9010 卡的工作原理决定的）。如果这种误差能满足用户要求，也就能实现不同板间的插补。

还有一种方法，用函数 LM_IOOut_9010 (….)、LM_Wait_I_9010 (….)实现两板的同步。在两块卡的端子板上连一根线，比如板 0 的 1 号输出点连在板 1 的 1 号输入点上，在两卡的插补缓冲区中分别写入如下命令：

```
….
LM_IOOut_9010 (0，1，1，100)；//板 0 的 1 号输出点为 1
LM_Line_9010 (0，….);              //执行线性插补
….

LM_Wait_I_9010 (1，1，100)；      //等待板 1 的 1 号输入点为 1 时（实际为从板 0 发出）
LM_Line_9010 (1，….);               //执行线性插补
….
```

在上例中，两块板的线性插补同步时间在 0.5 毫秒之内。

（9）如何运用 PWM 输出与插补速度随动命令

在激光加工中，PWM 信号一般用来控制激光功率，**PWM 输出与插补速度随动命令**（函

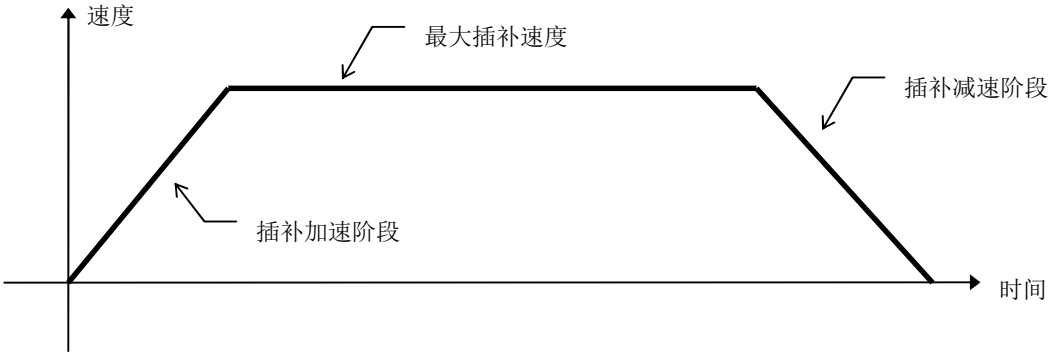
数 LM_PWM_9010）用于实现这个目标。在程序前端写入 LM_PWM_9010 命令，往后 PWM 脉冲输出的占空比将随插补速度按比例线性变化，这些工作完全由板上的 DSP 完成，用户使用起来将非常方便。

（10）插补结束函数（LM_End_9010）起什么作用

插补结束函数（LM_End_9010）最主要的作用有两个：1、告诉 9010 卡，当执行到 **插补结束函数（LM_End_9010）**所在行时，插补速度为 0，即停止插补运动。2、告诉 9010 卡，何时开始新的插补运动，等待新的插补开始命令 **LM_Start_9010（）**。

如前所述，9010 卡对写入插补缓冲区的命令进行插补预处理，对各个插补线段的起点位置速度及终点位置速度进行预先计算，比如：当只执行一条很简单的直线插补命令 LM_Line_9010(0,1000,0,0,0,1000,100)时，其后也应该加入插补结束函数（LM_End_9010），这时 9010 明白只有一条直线插补，该线段终点位置的插补速度为 0，如下：

```
……
LM_Line_9010(0,1000,0,0,0,1000,100);
LM_End_9010（0）;
LM_Start_9010（0,0）;
```



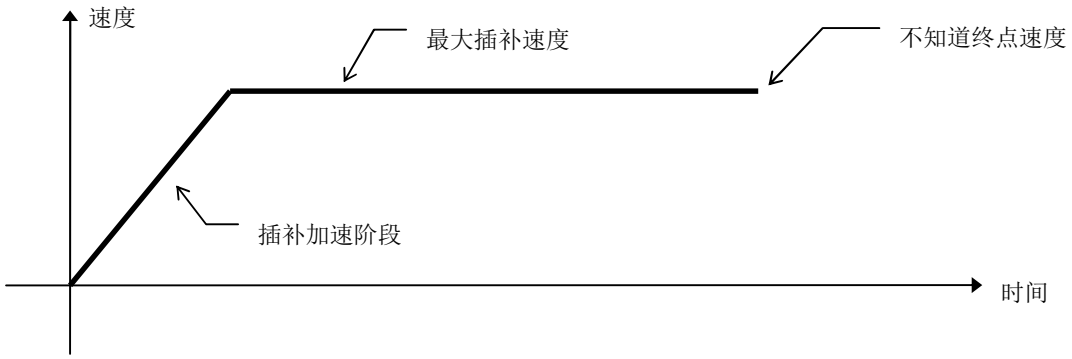
插补速度曲线

如果不写插补结束函数（LM_End_9010），例如下：

```
……
LM_Line_9010(0,1000,0,0,0,1000,100);
LM_Start_9010（0,0）;
```

由于不知道该线段终点位置的插补速度到底是多少，9010 卡不能完成插补预处理（即不能完成该线段的速度规划），在上例中，在发出插补开始命令后，实际 9010 卡并不立即执行插补运动，除非插补开始命令代入强制执行参数，如下：

```
LM_Line_9010(0,1000,0,0,0,1000,100);
LM_Start_9010（0,1）;          //第 2 个参数为 1，表示强制执行（9010 卡认为
                               //该线段的终点速度为 0）。
```



未完成的插补速度曲线

需要指出的是，能告知 9010 卡插补终点速度为 0 的还有以下几个函数：

插补等待命令：	short	LM_Wait_9010 (….);
插补等待输入点命令：	short	LM_Wait_I_9010 (….);
插补输出 IO 点命令：	short	LM_IOOut_9010 (….);
插补输出 PWM 命令：	short	LM_PWM_9010 (….);

（11）插补输出 IO 点函数 LM_IOOut_9010 与普通输出 IO 点函数 WriteIo_9010 的关系

在插补正在执行阶段（用函数 LM_GetState_9010 检测到的状态值为 7 ），函数 LM_IOOut_9010 的优先级高于普通输出 IO 点函数 WriteIo_9010，但低于按位输出 IO 点函数 WriteIoBit_9010。如，在插补序列中有函数 LM_IOOut_9010，它对第 1 号输出点置为 1，当插补运动执行到该行时，1 号输出点为 1，而这时再用 WriteIo_9010 函数对 IO 点置位时，第 1 号 IO 点的输出将不受影响，除非插补状态值不为 7。而按位输出 IO 点函数 WriteIoBit_9010 可随时控制输出点。

（12）函数 LM_GetState_9010 说明

函数 LM_GetState_9010 的原意是获得插补状态，其返回值有以下几种:

0: 停止状态
1: 被轴停止命令结束 (建议用 Abort_9010 命令)
2: 被行结束停止命令停止
3: 被 LM_End_9010 结束
4: 插补缓冲区空
5: 被进给倍率为 0 暂停
6: 被进给暂停挂起
7: 插补正在进行

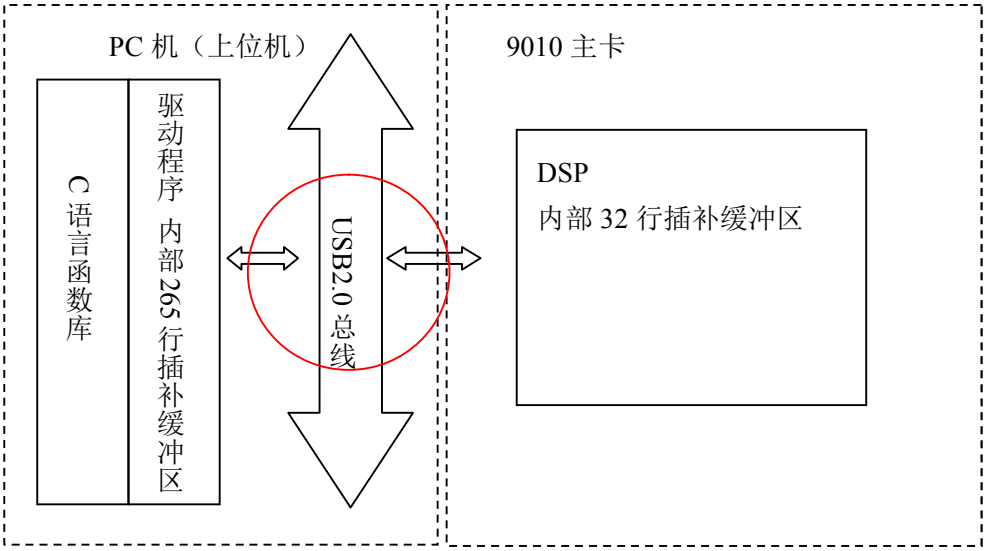
255:　　命令不成功

其中返回值 4 有两层意思：

1、插补缓冲区空

2、等待后续命令

9010 卡采用如下策略加大插补缓冲区:



即: 在上位机的驱动程序中有 265 行插补缓冲区（缓冲区 1）、在 9010 卡内部有 32 行插补缓冲区（缓冲区 2）。

例如：C 语言编程片断如下：

```
short IpoL_State_9010;
short len;

    len=LM_GetBuffLen_9010(0);

IpoL_State_9010=LM_GetState_9010(0);

if(IpoL_State_9010==4 && len!=256)     //插补缓冲区空但在插补缓冲区中还有命令
{
    //产生的原因是：插补命令滞留在驱动程序中的 265 行缓冲区中，而 9010 卡内部的
    //32 行缓冲区是空的
```

```
        Sleep(1);               //暂停 1 毫秒
        len=LM_GetBuffLen_9010(0);
        IpoL_State_9010=LM_GetState_9010(0);    //再次检测插补状态
        if(IpoL_State_9010==4 && len!=265)     //插补缓冲区空但在插补缓冲区中还有命令
        {
            LM_Start_9010(0,1);   //强制执行
        }
        else
        {
            ……
        }
    }
```

当插补缓冲区还滞留有插补命令，而函数 LM_GetState_9010 返回值为 4 时（等待后续命令），其原因有两种：

1. 插补命令滞留在驱动程序中的 265 行缓冲区中，而 9010 卡内部的 32 行缓冲区是空的。在执行函数 LM_GetBuffLen_9010 时，该函数本身将根据情况会自动完成插补命令从缓冲区 1 到缓冲区 2 的转移。如果实际产生转移，则插补状态值将改变，所以在上例程序片断中，让程序暂停 1 毫秒，再次检测插补状态。

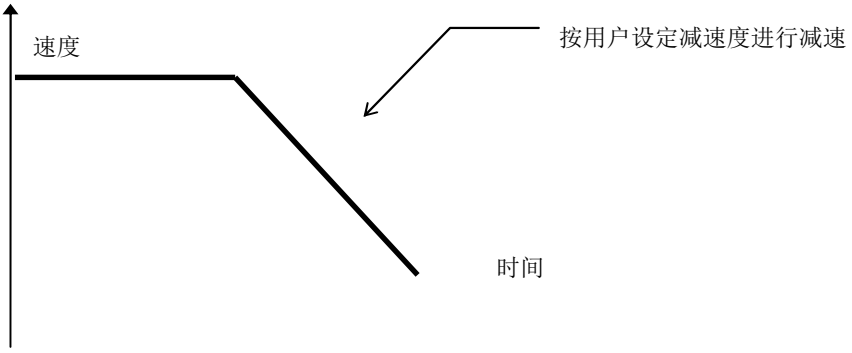
2. 就是问题(10)中所阐述的现象，即 9010 卡未能完成插补预处理，没有完成该线段的插补速度规划。解决方法是加入插补结束函数（LM_End_9010）或 用 函数 LM_Start_9010（0,1）强制运行。

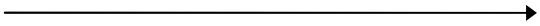
§2.5.9 其它说明

⑴轴停止命令

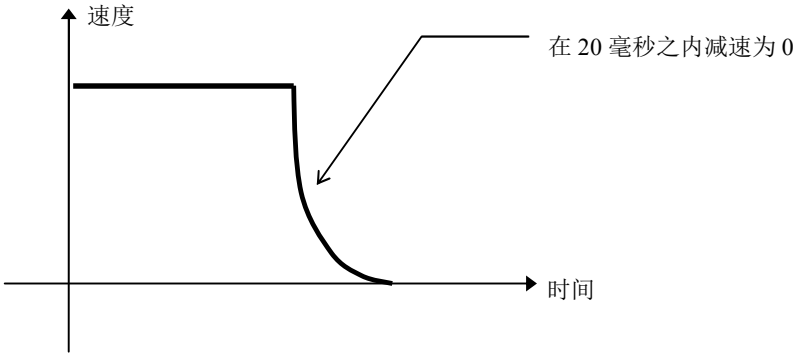
轴停止命令有三种，它们的速度曲线如下：

1． StopAxis_9010 模式：

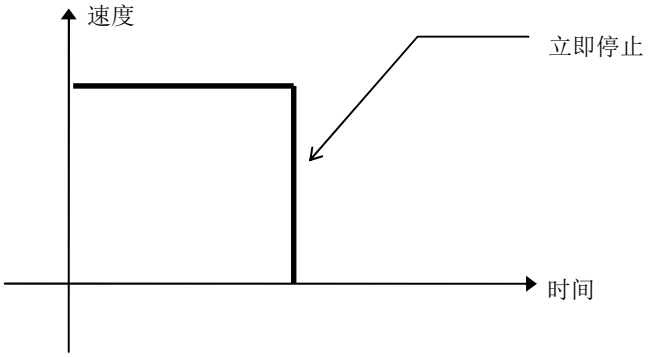




2．AbortAxis_9010 模式：



3．CeaseAxis_9010 模式：



基于 9010 卡的工作原理，如果编程如下：

```
…. ….
CeaseAxis_9010（0，0）；          //0 轴立即停止
State= ReadAxisState_9010(0，0);   //读取 0 轴状态
…. ….
```

State 并不能马上返回-5（轴被 CeaseAxis_9010 命令停止），一般按如下编程，State 就能返回-5 值。

```
…. ….
CeaseAxis_9010（0，0）；          //0 轴立即停止
Sleep（1）；                      //程序等待 1 毫秒
State= ReadAxisState_9010(0，0);   //读取 0 轴状态
…. ….
```

⑵通用 IO 点

9010 卡有 8 个输出点，20 个输入点，其中 20 个输入点中第 9 至第 20 点有双重目的，

在缺省状态下，它们分别是 4 个轴的正限位输入点、0 位输入点、负限位输入点。可用函数 SetAxisIO_9010 来控制它们是否有效，无效的轴 IO 可以当普通 IO 来用。当轴 IO 有效时，当碰到限位时，输入点会以中断的方式通知 DSP，DSP 将以 AbortAxis_9010 模式使轴停止运动。9010 还可接入急停开关，用 Set_Emergency_Stop_9010 函数可以使急停开关有效并与第 1 至第 8 某一输入点对应。缺省时急停开关无效。当急停信号到来时，DSP 会以 CeaseAxis_9010 模式使全部轴停止。

⑶PWM/DA 输出与编码器输入

9010 卡用同一个输出信号来实现 PWM 或 DA 输出，当用户选择 DA 输出时，在信号前端还是 PWM 信号，它的频率是固定的 25K，9010 卡通过改变占空比来改变电压。

在实际应用中，PWM 输出可控制激光电源，DA 输出可控制主轴转速。编码器输出可接一个手摇轮。

⑷轴位置计数器溢出

9010 卡的轴位置计数器在主卡上的 FPGA 中，是一个 32 位可逆计数器，它真实地记录每个轴发出的脉冲个数，其有效值范围为：-2147483648 至 2147483647。当轴沿一个方向长时间运动时，轴位置计数器有可能溢出。每转 10000 线的伺服电机，按 3000 转/分钟旋转，当运行大概不到 72 分钟时，位置计数器就溢出了。

9010 卡中，如果轴位置计数器溢出，当轴以位置模式或插补模式运行时，9010 卡会以立即停止模式（CeaseAxis_9010 模式）让轴停止运行，同时报警（用 GetErrorNo_9010 函数可检测到）；如果是速度模式（用 StartAxisVel_9010 命令启动轴运动），则 9010 卡只会报警，不会将轴停止。

溢出时的轴位置计数器将从最大正值转到最小负值，或从 最小负值转到最大正值。

§ 2.5.10 库函数一览表

表 2.5 库函数一览表 (C 语言方式)

编号	函数名	描述	
初始化函数			
1	InitCard_9010	初始化 9010 卡	
23	ExitCard_9010	退出 9010 卡,并释放所占资源	
轴常规设定函数			
119	GetAxisMode_9010	读轴用户设定模式	
98	SetAxisWorkMode_9010	设置轴工作模式	
15	SetAxisIO_9010	设置轴 IO 是否有效	
17	SetAxisOutMode_9010	设置轴的输出模式	
轴闭环控制函数			

99	SetAxisKP_901	设置轴 PID 调节比例系数	
100	SetAxisKI_901	设置轴 PID 调节积分系数	
101	SetAxisKD_901	设置轴 PID 调节微分系数	
102	SetAxisIL_901	设置轴 PID 调节积分限	
112	SetAxisOPC_901	设置轴电压输出 0 点补偿	
	SetAxisFVRate_901	设置轴 PID 调节速度前馈系数	
103	SetAxisFV_901	设置轴 PID 调节速度前馈	
104	SetAxisPEL_901	设置轴位置闭环跟随误差极限	
123	SetEncoderX4X1_9010	设置轴 ***位置反馈编码器*** 和 ***附加编码器*** 的内部倍频数	
轴状态读取函数			
	ReadAxisTheoryPos_9010	读取轴的当前理论位置	
10	ReadAxisEncodePos_9010	读取轴的编码器位置	
2	ReadAxisPost_9010	返回轴当前实际位置	
	ReadAxisTECV_9010	读取轴螺距误差补偿数据	
26	ReadAxisVel_9010	返回轴的当前速度	
27	ReadAxisState_9010	读取轴的状态	
1021	IsPosCatch_9010	读取轴位置捕获标志位	
124	ResetPosCatch_9010	轴位置捕获标志位复位	
1022	ReadCatchPos_9010	读取轴的捕获位置	
轴螺距误差补偿设置函数			
73	SetAxisTEC_9010	设置轴螺距反向间隙补偿	
111	SetAxisTECData_9010	设置轴螺距误差补偿数据	
	SetAxisTECWork_9010	启动/停止 轴螺距误差补偿	
单轴运动控制函数			
4	SetAxisPos_9010	设置轴的位置	
5	SetAxisVel_9010	设置轴的速度	
88	SetAxisStartVel_9010	设置轴的起跳速度	
95	SetAxisStopVel_9010	设置轴的停止速度	
6	SetAxisAcc_9010	设置轴的加速度	
96	SetAxisDec_9010	设置轴的减速度	
7	StartAxis_9010	轴开始运行,位置模式	

8	StartAxisVel_9010	设置轴速度模式,并按所设速度开始运行	
9	StopAxis_9010	轴停止,按所设加速度减速停止	
19	AbortAxis_9010	轴停止,在 10 毫秒之内减速停止	
20	CeaseAxis_9010	轴停止,立即停止,无减速过程	
24	GoHome_9010	轴回 Home 点	
	SetAxisSAcc_9010	设置轴 S 型加速度(保留函数)	
3	Home_9010	轴位置清零	
11	HomeFB_9010	轴位置编码器清零	
114	LookZIndex_9010	轴找一转脉冲(轴回 Home 点)	
24	SetAxisOffset_9010	设置轴位置偏移值	
109	SetAxiFBOffset_9010	设轴位置编码器偏移值	
电子齿轮与位置跟随			
75	StartAxisEGear_9010	设置轴电子齿轮运动	
121	SetActiveEncoder_9010	在轴跟随编码器运动之前,设主动编码器号	
74	SetAxisFE_9010	设置轴跟随编码器运动	
76	CancelAxisFEG_9010	取消轴跟随运动	
80	GetEnAuto0Flag_9010	获得主动编码器自动清零标志	
79	ResetEn0Flag_9010	主动编码器自动清零标志置 0	
轴其它控制函数			
105	SetAxisDAOut_9010	设置轴输出电压	
106	SetAxisPWMOut_9010	设置轴 PWM 脉冲输出	
107	AxisPWMStop_9010	轴 PWM 频率输出停止	
110	SetAxisMotorOnOff_9010	设轴电机 On 或 Off，使能	
附加编码器控制函数			
10	ReadEncoderPos_9010	读取编码器位置	
11	HomeEncode_9010	复位编码器,编码器位置清零	
113	SetEncodeCount_9010	设附加编码器位置计数器初值	
PWM 脉冲/DA 输出控制			
12	PwmOut_9010 PwmOut2_9010	PWM 脉冲输出	
13	PwmStop_9010	PWM 脉冲停止输出	
68	DAOut_9010	DA(数模转换)模拟量输出	

通用 IO 读写函数			
29	Set_Emergency_Stop_9010	设置 IO 急停信号	
14	ReadIO_9010	读取通用输入点 I1-I20 状态	
	ReadIOBit_9010	按位读取通用输入点 I1-I20 状态	
16	WriteIo_9010	设置通用输出点 O1-O8 状态	
81	WriteIoBit_9010	按位设置通用输出点 O1-O8 状态	
82	ReadOs_9010	读取输出点 O1-O8 状态	
83	ReadOsBit_9010	按位读取输出点 O1-O8 状态	
118	ReadMPGIO_9010	读取手轮 IO	
插补函数			
插补初始化函数			
30	LM_INIT_IPOL_ENGINE	初始化插补引擎	
31	LM_EXIT_IPOL_ENGINE	退出插补引擎	
32	LM_SetXAxis_9010 LM_SetYAxis_9010 LM_SetZAxis_9010 LM_SetWAxis_9010	将实际轴与插补引擎的 X 轴相匹配	
33	LM_OffXAxis_9010 LM_OffYAxis_9010 LM_OffZAxis_9010 LM_OffWAxis_9010	撤消插补引擎的 X 轴匹配	
54	LM_GetXStartPos_9010 LM_GetYStartPos_9010 LM_GetZStartPos_9010 LM_GetWStartPos_9010	获得插补轴当前位置,也是轴插补的开始位置,在发送插补命令 (LM_Line_9010,IpolArc_9010)开始前调用	
插补顺序执行函数			
42	LM_Line_9010	直线插补 该命令将把插补数据送入 9010 卡的插补缓存器中,9010 卡的插补缓存器总共有 16 行	
	LM_LineMaxV_9010		
65	LM_LineMeasure_9010	直线插补并测量输入点	
66	LM_GetMeasureState_9010	获得插补测量状态	
77	LM_LineFE_9010	直线插补 跟随编码器位置	
43	LM_ArcCW_9010	圆弧插补,顺圆	

		该命令将把插补数据送入 9010 卡的插补缓存器中,9010 卡的插补缓存器总共有 16 行	
	LM_ArcCW_YZ_9010		
	LM_ArcCW_ZX_9010		
44	LM_ArcCCW_9010	圆弧插补,逆圆 该命令将把插补数据送入 9010 卡的插补缓存器中,9010 卡的插补缓存器总共有 10 行	
	LM_ArcCCW_YZ_9010		
	LM_ArcCCW_ZX_9010		
60	LM_Wait_9010	插补等待	
61	LM_PWM_9010	插补模式的 PWM 输出，占空比随速度变化	
62	LM_IOOut_9010	插补模式的 IO 点输出	
63	LM_Wait_I_9010	插补等待通用 IO 点输入	
45	LM_End_9010	插补结束 该命令将把插补数据送入 9010 卡的插补缓存器中,9010 卡的插补缓存器总共有 10 行 当插补引擎运行到该行时将停止运行,如要重新开始,则再送入插补数据命令 (LM_Line_9010,LM_ArcCW_9010,LM_ArcCCW_9010),并发 LM_Start_9010 命令	
插补控制函数			
40	LM_SetACCDec_9010	设置插补加速度和减速度	
71	LM_SetSAccPower_9010	设置插补 S 型加速度(1,2,3,4)指数(保留函数)	
41	LM_Start_9010	插补开始	
46	LM_CleanBuff_9010	清除插补缓存	
55	LM_GetBuffLen_9010	获得插补缓存剩余长度	
47	LM_SetMinVel_9010	设插补最小速度	
57	LM_SetMaxVel_9010	设插补最大速度	
48	LM_SetParaAngle_9010	设插补参数,角度	
49	LM_GetLineNO_9010	获得插补当前行号	
1002	LM_GetFactVel_9010	获得实际插补速度	
	LM_SetPauseTime_9010	设置插补暂停停止时间	
50	LM_Pause_9010	插补暂停	
51	LM_Resume_9010	恢复插补暂停	
52	LM_SetSpeedRate_9010	设插补速率	
53	LM_LineEnd_9010	插补运行完当前行停止	
56	LM_GetState_9010	获得插补状态	

58	LM_SetSpeedPri_9010	设插补速度优先还是精度优先	
59	LM_SetSysPara_9010	设插补系统参数	
87	LM_SetAxisMaxErrLtd_9010	设轴最大插补位置误差限制	
78	LM_GetFEnState_9010	获得轴编码器跟随已达到目标状态	
Nurbs 曲线插补函数			
1000	LM_SetNurbsScanMode_9010	设置 Nurbs 曲线插补预处理的扫描模式	
1001	LM_SetNurbsVelCtrl_9010	设置 Nurbs 曲线插补时的速度控制	
93	LM_SetNurbsAccDec_9010	设置 Nurbs 曲线插补加速度和减速度	
97	LM_SetNurbsCompCoef_9010	设 Nurbs 曲线插补误差补偿系数	
90	LM_NurbsInit_9010	初始化 Nurbs 曲线,为 Nurbs 曲线插补做准备	
91	LM_NurbsData_9010	设 Nurbs 曲线插补数据	
92	LM_Nurbs_9010	Nurbs 曲线插补	
	LM_Nurbs4Axis_9010	4 轴 Nurbs 曲线插补	
1013	LM_SendNurbsData_9010	向 9010 卡传输 NURBS 曲线数据	
1012	LM_GetNubrsExecPara_9010	读取 9010 卡 NURBS 曲线插补运行参数值	
	LM_GetNurbsInBuffLen_9010	获得 Nurbs 曲线滞留在插补缓冲区的数	
1010	LM_GetNurbsErrorNo_9010	获得产生 Nurbs 曲线(插补)的错误号	
Nurbs 曲线计算函数			
1004	Set_NurbsInit_9010	初始化 Nurbs 曲线,为 Nurbs 曲线计算做准备	
1005	Set_NurbsData_9010	设 Nurbs 曲线计算数据	
1006	Set_NurbsEnd_9010	设 Nurbs 曲线数据结束	
1007	Get_NurbsPos_9010	计算 Nurbs 曲线轴位置(型值点)坐标	
1008	Get_NurbsPosVB_9010	计算 Nurbs 曲线轴位置(型值点)坐标(VB 风格)	
1009	Get_NurbsLen_9010	计算 Nurbs 曲线长度	
1011	Get_NurbsErrorNo_9010	获得产生 Nurbs 曲线(计算)的错误号	
94	LM_SetForceCtrl_9010	设置插补力控制模式	

读取 9010 卡有关版本信息			
28	ReadFirmwareVersion_9010	读取 9010 控制卡固件版本号	
67	ReadDllVersion_9010	读取 9010 控制卡动态链接库版本号	
70	GetDriverVersion_9010	读取 9010 卡的驱动版本号	
	GetErrorNo_9010	获得错误号	
CAN 总线扩展			
2000	RegCANExp_9010	登记 CAN 总线扩展卡	
116	EnableCANExp_9010	CAN 总线扩展卡使能(初始化邮箱)	
117	SendCANData_9010	CAN 总线扩展卡发送数据 (邮箱发送消息)	
2001	ReadCANL_9010	读取 CAN 总线扩展卡数据 低 4 字节	
	ReadCANH_9010	读取 CAN 总线扩展卡数据 高 4 字节	
	GetCANErrorNo_9010	获得 CAN 总线错误号	

§2.6 NURBS 曲线

由于 Nurbs 曲线的特殊性，在本节将单独对它进行讲解。

§2.6.1 NURBS 曲线概论

NURBS 曲线是 ***非均匀有理 B 样条*** 曲线（Non Uniform Rational B-spline）的英文缩写，其数学定义为：

$$P(K)=\frac{\sum_{i=0}^{n}N_{i,m}(K)R_iP_i}{\sum_{i=0}^{n}N_{i,m}(K)R_i}\quad ,\quad P(K)\text{为曲线上的位置向量，}\ N_{i,m}(K)\text{为}m\text{次样条基函数}$$

基函数由递推公式定义：

$$N_{i,0}(K)=\begin{cases}1(K_i\le K\le K_{i+1})\\0(\text{其它})\end{cases}$$

$$N_{i,m}(K)=\frac{(K-K_i)N_{i,m-1}(K)}{K_{i+m}-K_i}+\frac{(K_{i+m+1}-K)N_{i+1,m-1}(K)}{K_{i+m+1}-K_{i+1}},\quad m\ge 1$$

式中：P_i 为控制点；R_i 为权因子；K 为节点矢量

NURBS 曲线有诸多优点，比如：局部修改性、多阶导数连续性、投影变换不变性等，作为表示自由曲线的一种方法，NURBS 曲线已经得到了广泛的使用，在目前流行的 CAD/CAM 系统中，曲线、曲面的实体造型绝大部分都是用 NURBS 曲线、曲面表示，而在生成加工轨迹时，由于当前大部分 CNC 系统(计算机数控系统)不支持 NURBS 曲线加工，所以只好用短直线来近似。这种近似带来了精度、加工效率损失，和短直线插补相比, NURBS 曲线插补加工, 加工表面非常接近设计要求的轮廓曲面，且加工代码可在一组程序段中指定, 可大大减少程序的大小。由于 3 次 NURBS 曲线 2 阶导数的连续性，其加工速度和加工精度都是短直线不能比拟的。

9010 卡有以下有关 NURBS 曲线函数：

设置 Nurbs 曲线插补预处理的扫描模式：

short LM_SetNurbsScanMode_9010(….);

设置 Nurbs 曲线插补时的速度控制：

short LM_SetNurbsVelCtrl_9010(….);

设置 Nurbs 曲线插补加速度和减速度：

short LM_SetNurbsAccDec_9010(….);

设 Nurbs 曲线插补误差补偿系数：

short LM_SetNurbsCompCoef_9010(….);

读取 NURBS 曲线插补运行参数值:

float LM_GetNubrsExecPara_9010(….);

获得在插补缓冲区中有几条 Nurbs 曲线：

short LM_GetNurbsInBuffLen_9010(….);

获得产生 Nurbs 曲线插补错误的编号:

short LM_GetNurbsErrorNo_9010(….);

向 9010 卡转输 NURBS 曲线数据:

short LM_SendNurbsData_9010(….);

NURBS 曲线插补命令：

初始化 Nurbs 曲线： short LM_NurbsInit_9010 (….);

设 Nurbs 曲线插补数据： short LM_NurbsData_9010(….);

Nurbs 曲线插补命令（3 轴）： short LM_Nurbs_9010(….);

Nurbs 曲线插补命令（4 轴）： short LM_Nurbs4Axis_9010(….);

有关 NURBS 曲线计算的函数：

初始化 Nurbs 曲线,为 Nurbs 曲线计算做准备:

short Set_NurbsInit_9010(….);

设 Nurbs 曲线计算数据: short Set_NurbsData_9010(….);

设 Nurbs 曲线数据结束:　　short　Set_NurbsEnd_9010(….);

计算 Nurbs 曲线轴位置(型值点)坐标:

short　Get_NurbsPos_9010(….);

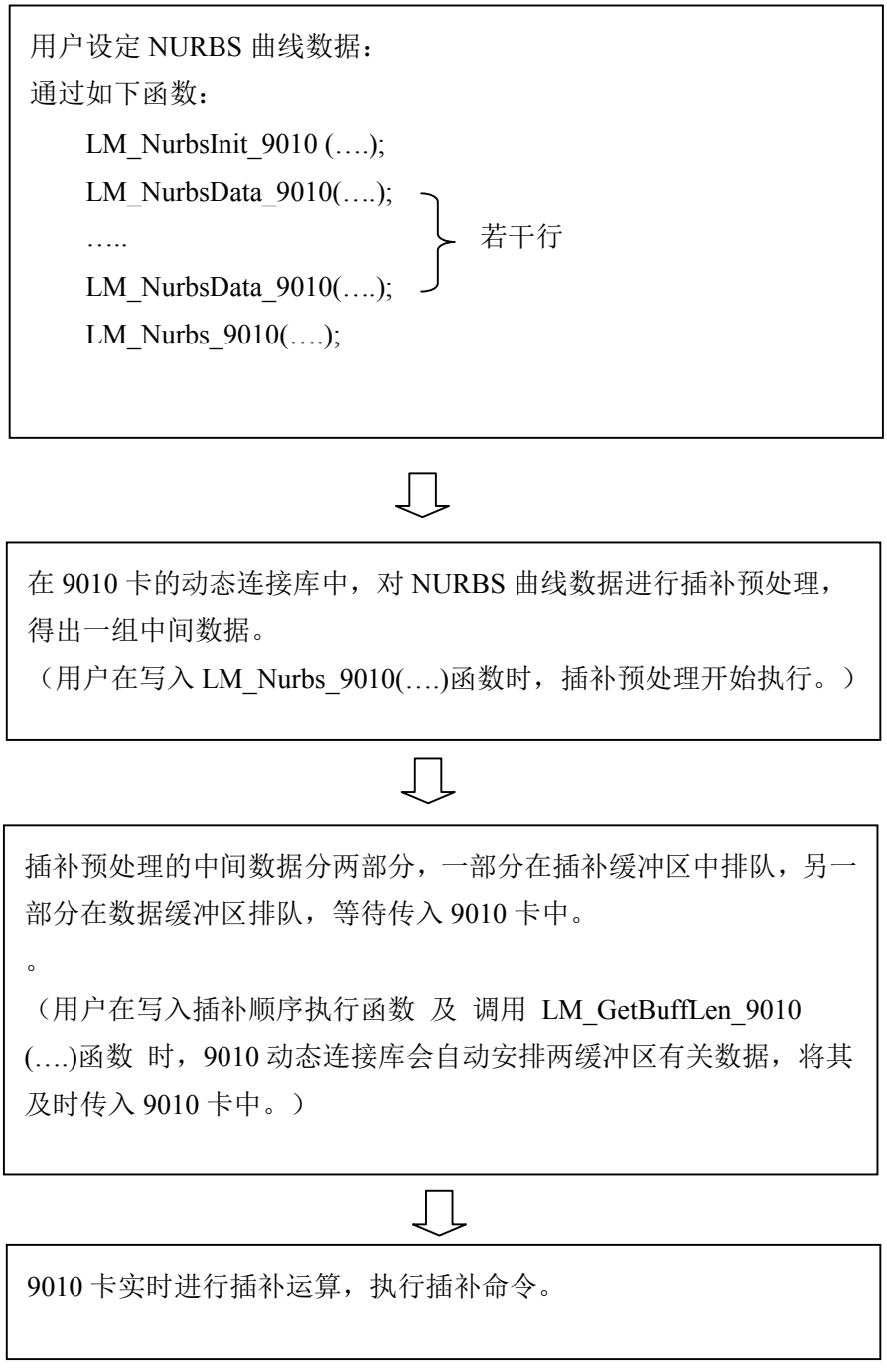
计算 Nurbs 曲线长度:　　double Get_NurbsLen_9010(….);

获得产生 Nurbs 曲线计算错误的编号:

short　Get_NurbsErrorNo_9010(….);

§2.6.2　NURBS 曲线插补工作流程

针对 NURBS 曲线，9010 卡工作过程如下：

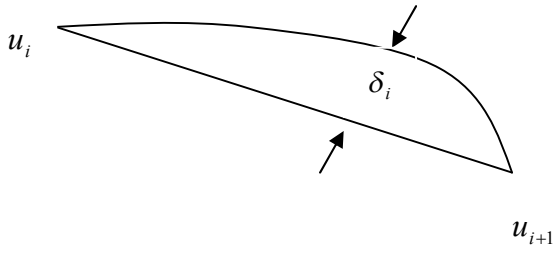


§2.6.3 精度与速度控制

为了控制 NURBS 曲线的插补精度及运行平稳性，引入以下概念：

弦高误差：

在 NURBS 曲线插补中，每个插补周期都是一个直线代替曲线弧的处理过程，同时引入弦高误差δ，如图所示。



弦高误差δ　　原理图

弦高误差δ的大小与进给速度ΔV、曲率半径ρ有关。如果以圆弧段近似代替曲线弧，则可得 NURBS 曲线插补的弦高误差关系式：

$$\Delta V_i = \frac{\sqrt{4\delta_i(2\rho-\delta_i)}}{T} \approx \frac{\sqrt{8\rho_i\delta_i}}{T}$$

其中：T 为插补运算周期，通常情况下，ρ〉〉δ。

从上式可以看出弦高误差δ与进给速度ΔV_i、曲率半径ρ密切相关。

为了将整个 NURBS 曲线插补轨迹的弦高误差限制在指定的允许误差范围之内，给定许用**最大弦高误差**$\delta_{\max}$。

法向加速度：

在对 NURBS 曲线进行插补处理时，不但要考虑沿 NURBS 曲线轨迹切线方向的加速度，还要考虑轨迹法向加速度。

法向加速度的计算公式为：

$a = \frac{v^2}{\rho}$　　其中：v 切向速度，ρ 为曲率半径。

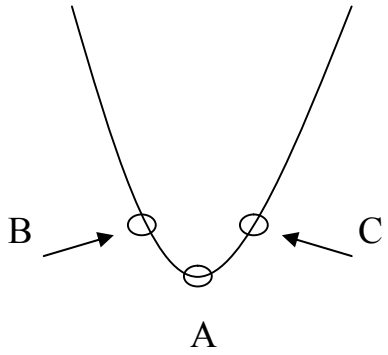
由上式可以看出，在进给速度较大、轨迹曲率半径较小的情况下，法向加速度会很大，由物理学公式

F=ma

可知，法向加速度过大，机床所受力 F（沿 NURBS 曲线的法向）可能会超出机床允许的范围之内，因此有必要对 NURBS 曲线的法向加速度进行限制。

Nurbs 曲线插补预先扫描：

从上述可知，“弦高误差”和“法向加速度”与加工速度、曲率半径密切相关，为了控制“弦高误差”和“法向加速度”，在插补前，9010 卡会对 NURBS 曲线进行插补预处理，其过程是按一定速度对整条 NURBS 曲线进行逐点计算（对 Nurbs 曲线全长进行扫描），找出曲率半径的极低点位置，及相关的减速点及加速点位置（如下图 A、B、C 点），在实际 NURBS 曲线插补时，系统将在到达曲率半径极低点（A 点）前的 B 点时减速，在离开 A 点后，从 C 点起速度随曲率半径增大而加速，直至加速到正常加工速度。



基于以上原因，函数 LM_SetNurbsScanMode_9010(….)、LM_SetNurbsVelCtrl_9010(….)将完成以上参数(弦高误差、法向加速度、预扫描速度)设定。

由于 Nurbs 曲线插补的特殊性，函数 LM_SetNurbsAccDec_9010(….)将单独给 Nurbs 曲线插补设插补加速度和插补减速度。

§2.6.4　编程说明

由 NURBS 曲线的定义和性质可知：

当节点矢量数为（m+1）, 幂次为 p，控制顶点数为（n+1）时，m, p 和 n 三者之间的关系为

m=n+p+1。

组成一个最简单的 3 次 NURBS 曲线必须至少有 4 个控制点，所以，3 次 NURBS 曲线另外还有多出的 4 个节点数据，所以在组成一条完整的 NURBS 曲线，必须按如下方式书写：

```
LM_NurbsInit_9010 (….);
LM_NurbsData_9010(….);
…..                          } 若干行，最少 4 行
LM_NurbsData_9010(….);
LM_Nurbs_9010(….);……本函数将代入另 4 个节点数据
```

如果产生异常情况，可调用函数 Get_NurbsErrorNo_9010(….)获得产生错误的原因。

§2.6.5 NURBS 曲线的计算

为了编程方便，9010 卡给出了有关 NURBS 曲线坐标位置计算和曲线长度计算的函数。这些函数可单独使用，与插补无关。

设 NURBS 曲线数据：

初始化 Nurbs 曲线,为 Nurbs 曲线计算做准备:

short Set_NurbsInit_9010(….);

设 Nurbs 曲线计算数据: short Set_NurbsData_9010(….);

设 Nurbs 曲线数据结束: short Set_NurbsEnd_9010(….);

计算 Nurbs 曲线轴位置(型值点)坐标:

short Get_NurbsPos_9010(….);

计算 Nurbs 曲线长度:

double Get_NurbsLen_9010(….);

在函数 Get_NurbsPos_9010(….)、Get_NurbsLen_9010(….)中，都有 **参数变量** Up，Up 值的范围是: 0 - (控制点数-3)，比如，Nurbs 曲线由 4 个控制点组成，则：

Up：0 – 1.0

Nurbs 曲线由 7 个控制点组成，则：

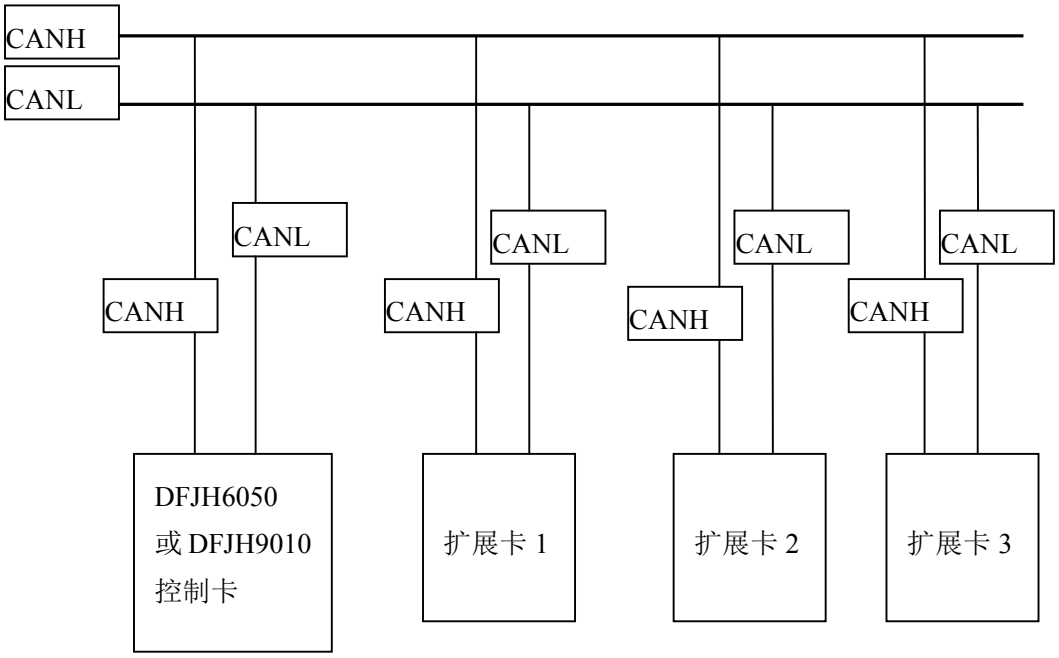
Up：0 – 4.0

在 9010 卡中，Nurbs 曲线在内部以矩阵方式表示（详细说明请看本书附录 A 的有关内容），Up 就是矩阵行列式中的变量系数。

§2.7 CAN 总线扩展

9010 卡有 CAN 总线扩展接口，我公司目前 CAN 总线 IO 扩展板有两种型号，分别为电压输出型 E1044（16 入/16 出）和继电器输出型 E1044_RELAY（16 入/16 出）。扩展卡支持 CAN2.0 协议，通讯速率 1Mbps，数据位 16 位，标准帧传输。由于 Can 总线的传输距离远，扩展板非常适合远程 IO 控制。

如果用户在应用 9010 卡时，需要扩展 IO 点，硬件连接如下：



CAN 总线可以使用普通单根细导线，或双绞线，传输速率为 1Mbps 时最远传输距离为 40m。

在软件方面，有以下函数：

RegCANExp_9010	***登记 CAN 总线扩展卡***
EnableCANExp_9010	***CAN 总线扩展卡使能***
SendCANData_9010	***CAN 总线扩展卡发送数据***
ReadCANL_9010	***读取 CAN 总线扩展卡数据 低 4 字节***
ReadCANH_9010	***读取 CAN 总线扩展卡数据 高 4 字节***

编程步骤：

1、 初始化：

```
RegCANExp_9010（）；//登记 CAN 总线扩展卡，用几块登记几块，用不同地址分开
RegCANExp_9010（）；
.....
RegCANExp_9010（）；
EnableCANExp_9010;        //使能 CAN 总线扩展卡
```

2、读写：

用 SendCANData_9010、 ReadCANL_9010、ReadCANH_9010 函数读写。

第三章　库函数详解

§3.1 初始化函数

§3.1.1　InitCard_9010

初始化控制卡

函数编号:1

函数原型：

C: short APIENTRY InitCard_9010(unsigned short Board_NO,short Axis0,short Axis1,short Axis2,short Axis3,short int OPC0,short int OPC1,short int OPC2,short int OPC3);

解释：

在调用函数库的各个函数之前，第一步首先必须对函数库进行初始化。

这个函数初始化电机控制卡，用户应在程序的初始化部分调用该函数，对每个板进行初始化。它将做以下工作：

①初始化板上芯片

②轴位置寄存器清零

③所有轴输出信号将根据硬件跳线方式（参看 ***§1.5 电机控制信号*** 节）及对应参数(Axis0、Axis1、Axis2、Axis3)值设置为特定模式，见下表。

④所有 IO 输出信号都设置为低电平（重新上电时）。如果不是重新上电，9010 卡将保持原来状态。

轴正在运行时是否可调用：否

参数：

board_NO:　　板号，可能值 0、1、2、3；一台计算机中最多可以运行 4 块控制卡。

Axis0,

Axis1,

Axis2,

Axis3　: 0-3:　　初始化对应轴输出模式:

0=自动模式;

1=方波脉冲输出模式(1);

2=单纯电压输出模式(4);

3=PWM 脉冲输出模式(5);

注：9010 卡的每个轴都可由 5 中工作模式，它们是

0=轴无效(0);

1=开环脉冲模式(1)

2=位置闭环脉冲输出模式(2)

3=位置闭环模拟量输出模式(3)

4=单纯电压输出模式(4)

5=单纯 PWM 脉冲输出模式(5)

参看 SetAxisWorkMode_9010 函数

OPC0

OPC1

OPC2

OPC3 : 对应轴在电压输出模式时的 0 点漂移补偿值.

跳线模式	Axis0-Axis3 值		解释
跨接： 	0	**自动模式**	**轴工作在单纯电压输出模式(4)** 轴控信号接口中模拟量输出信号 Vout、AGND **有效，**输出电压为理论 0.0V + 漂移补偿值 轴控信号接口中脉冲输出信号 PULS、/PULS、SIGN、/SIGN **无效，**所谓无效就是输出状态不可预知
	1	**方波脉冲输出模式**	**由于跳线跨接，实为非法，轴将工作在单纯电压输出模式(4)** 轴控信号接口中模拟量输出信号 Vout、AGND **有效，**输出电压为理论 0.0V + 漂移补偿值 轴控信号接口中脉冲输出信号 PULS、/PULS、SIGN、/SIGN **无效**
	2	**单纯电压输出模式**	**轴工作在单纯电压输出模式(4)** 轴控信号接口中模拟量输出信号 Vout、AGND **有效，**输出电压为理论 0.0V + 漂移补偿值 轴控信号接口中脉冲输出信号 PULS、/PULS、SIGN、/SIGN **无效。**
	3	**PWM 脉冲输出**	**轴工作在 PWM 脉冲输出模式(5)**

		模式	轴控信号接口中模拟量输出信号 Vout、AGND 无效，输出状态不可预知。 轴控信号接口中脉冲输出信号 PULS、/PULS、SIGN、/SIGN 有效。PULS、/PULS 将组成一对 PWM 脉冲输出信号。
不跨接： 	0	**自动模式**	**轴工作在开环脉冲模式(1)** 轴控信号接口中模拟量输出信号 **Vout、AGND 无效，**输出状态不可预知。 轴控信号接口中脉冲输出信号 **PULS、/PULS、SIGN、/SIGN 有效，**它们将组成电机位置脉冲的控制信号。
	1	**方波脉冲输出模式**	**轴工作在开环脉冲模式(1)** 轴控信号接口中模拟量输出信号 **Vout、AGND 无效，**输出状态不可预知。 轴控信号接口中脉冲输出信号 **PULS、/PULS、SIGN、/SIGN 有效，**它们将组成电机脉冲的控制信号。
	2	**单纯电压输出模式**	**跳线没有跨接，实为非法，轴将工作在开环脉冲模式(1)** 轴工作在开环脉冲模式 轴控信号接口中模拟量输出信号 **Vout、AGND 无效，**输出状态不可预知。 轴控信号接口中脉冲输出信号 **PULS、/PULS、SIGN、/SIGN 有效，**它们将组成电机脉冲的控制信号。
	3	**PWM 脉冲输出模式**	**跳线没有跨接，实为非法，轴将工作在开环脉冲模式(1)** 轴工作在开环脉冲模式 轴控信号接口中模拟量输出信号 **Vout、AGND 无效，**输出状态不可预知。 轴控信号接口中脉冲输出信号 **PULS、/PULS、SIGN、/SIGN 有效，**它们将组成电机脉冲的控制信号。

返回码：

0 或-1,-1=不成功,0=成功

参考函数：Exit9010Card

§3.1.2 ExitCard_9010

释放控制卡驱动库函数

函数原型：

函数编号:23

C： short APIENTRY ExitCard_9010(unsigned short Board_NO);

解释：

释放驱动函数库，这个函数释放 9010 函数库所占用资源。在用户程序退出之前应调用该函数一次。

轴正在运行时是否可调用：否

参数：

board_NO: 板号，可能值 0、1、2、3；

返回码：

(0) 成功

(-1) 不成功

参考函数：

InitCard_9010

§3.2 轴常规设定函数

§3.2.1 GetAxisMode_9010

读轴用户设定模式

函数编号:119

函数原型：

C: short APIENTRY GetAxisMode_9010(unsigned short Board_NO,unsigned short Axis_No);

解释：

这个函数将返回轴的硬件跳线设定状态。

轴正在运行时是否可调用：是

参数：

Board_NO　　卡号，可能值 0，1，2，3

Axis_No　　轴号，可能值 0，1，2，3

返回码：

0 或 1,　　对应轴的用户设定模式；

0=跳线跨接，轴工作模式只能为(0)、(3)、(4)、(5)模式。**参看 SetAxisWorkMode_9010 函数**

1=跳线不跨接，轴工作模式只能为(0)、(1)、(2)模式；**参看 SetAxisWorkMode_9010 函数**

-1=不成功,

参考函数：

SetAxisWorkMode_9010

§3.2.2　SetAxisWorkMode_9010

设置轴工作模式

函数编号:98

函数原型：

C：　short APIENTRY SetAxisWorkMode_9010(unsigned short Board_NO,unsigned short Axis_No,unsigned short WorkMode);

解释：

设置轴工作模式

硬件跳线（参看 ***§1.5 电机控制信号*** 节）决定函数 SetAxisWorkMode_9010 可设定范围，硬件跳线状态可用函数 GetAxisMode_9010 获得，当硬件跳线状态为 1 时（跳线不跨接），轴工作模式只能设 0、1、2 模式；为 0 时（跳线跨接），轴工作模式只能设 0、3、4、5 模式。

在插补引擎正在运行当中是否可调用：否

参数：

Board_NO : 0－3, 板号

Axis_No　: 0－3, 轴号

WorkMode : 设置轴工作模式:

0=轴无效(0)
1=开环脉冲模式(1)
2=位置闭环脉冲输出模式(2)
3=位置闭环模拟量输出模式(3)
4=单纯电压输出模式(4)
5=单纯 PWM 脉冲输出模式(5)

返回码：

0 或-1, -1=不成功, 0=成功

§3.2.3 SetAxisIO_9010

设置轴 IO 是否有效

函数编号:15

函数原型：

C: short APIENTRY SetAxisIO_9010(unsigned short Board_NO,unsigned short Axis_No,short PHN_flag,short Mode,short H_L_Act,short IO_index);

解释：

设置轴 IO 是否有效。每个轴可以设 4 个特定点，它们是：

正限位、Home 点、负限位、位置捕捉触发信号输入点

9010 卡初始化时轴 IO 无效。

轴正在运行时是否可调用：否

参数：

Board_NO : 0－3, 板号
Axis_No : 0－3, 轴号
PHN_flag: 正限位、Home 点、负限位、位置捕捉触发信号：
1=正限位；2=Home 点；3=负限位；4=位置捕捉触发信号；

Mode : 0、1、2 或 3,
0=该点无效；3=自由挂接模式；
1、2=对正负限位复用固定点中断模式；（保留功能）
1、2=对 Home 点固定点模式；（保留功能）
H_L_Act : 0 或 1, 0=该点低电平有效,1=该点高电平有效
IO_index: 1-16, 在 Mode=3 时（自由挂接模式）；该点挂接到通用输入点的那一点

位置捕捉触发信号的进一步解释：

当 PHN_flag=4（位置捕捉触发信号）时，Mode 值只能为：

0（无效）
1（有效，连续捕捉）
2（有效，等待 ***位置捕捉标志位*** 复位后 再继续捕捉）

且每个轴的 **位置捕捉信号输入点** 位置固定：0 轴对应 I17 点；1 轴对应 I18 点；2 轴对应 I19 点；3 轴对应 I20 点；

H_L_Act : 0=下降沿触发捕捉；1=上升沿触发捕捉

设捕捉点时，编程顺序很重要，应该在设轴工作模式函数 SetAxisWorkMode_9010 之后再设捕捉点。切记

返回码：

(0) 成功
(-1) 不成功

参考函数：

§3.2.4 SetAxisOutMode_9010

设置轴的输出模式

函数编号:17

函数原型：

C: short APIENTRY SetAxisOutMode_9010(unsigned short Board_NO,unsigned short Axis_No,short Mode_A,short Mode_B,short Mode_C);

解释：

这个函数用来设置轴脉冲的输出模式及轴方向，当轴的工作模式是：

1=开环脉冲模式;

2=位置闭环脉冲输出模式;

时，在 9010 板卡支持“脉冲/方向”和“正向脉冲/反向脉冲”两种输出模式。

轴正在运行时是否可调用：否

参数：

Board_NO	卡号，可能值 0，1，2，3
Axis_No	轴号，可能值 0，1，2，3
Mode_A	0 或 1， 0=共阳极输出,1=共阴极输出； （保留功能，设 1，因为是轴位置信号是差分输出）
Mode_B	0 或 1， 0=脉冲-方向模式输出,1=脉冲-脉冲模式输出
Mode_C	0 或 1， 0=轴方向正常,1=轴方向反转

返回码：

(0) 成功

(-1) 非法操作

§ 3.3 轴闭环控制函数

§ 3.3.1 SetAxisKP_9010

设置轴 PID 调节比例系数

函数编号:99

函数原型：

C： short APIENTRY SetAxisKP_9010(unsigned short Board_NO,unsigned short Axis_No,unsigned short Kp);

解释：

设置轴 PID 调节比例系数。

轴正在运行时是否可调用：是

参数：

Board_NO : 0－3, 板号
Axis_No : 0－3, 轴号

Kp : PID 调节比例系数; 范围: 整型数(0-65535); 单位: 无。

返回码：

0 或-1, -1=不成功, 0=成功

§3.3.2 SetAxisKI_9010

设置轴 PID 调节积分系数

函数编号:100

函数原型：

C: short APIENTRY SetAxisKI_9010(unsigned short Board_NO,unsigned short Axis_No,unsigned short Ki);

解释：

设置轴 PID 调节积分系数

轴正在运行时是否可调用：是

参数：

Board_NO : 0－3, 板号
Axis_No : 0－3, 轴号

Ki : 轴 PID 调节积分系数; 范围: 整型数(0-65535); 单位: 无。

返回码：

0 或-1, -1=不成功, 0=成功

§3.3.3 SetAxisKD_9010

设置轴 PID 调节微分系数

函数编号:101

函数原型：

C： short APIENTRY SetAxisKD_9010(unsigned short Board_NO,unsigned short Axis_No,unsigned short Kd);

解释：

设置轴 PID 调节微分系数

轴正在运行时是否可调用：是

参数：

Board_NO : 0－3, 板号
Axis_No : 0－3, 轴号

Ki : 轴 PID 调节微分系数; 范围: 整型数(0-65535); 单位: 无。

返回码：

0 或-1, -1=不成功, 0=成功

§3.3.4 SetAxisIL_9010

设置轴 PID 调节积分限

函数编号:102

函数原型：

C： short APIENTRY SetAxisIL_9010(unsigned short Board_NO,unsigned short Axis_No,unsigned short IL);

解释：

设置轴 PID 调节积分限，当轴的 PID 调节器中积分项大于该设定值时，积分项的计算输出值将为 0，其目的是为了减小系统过调。

轴正在运行时是否可调用：是

参数：

Board_NO : 0－3, 板号
Axis_No　: 0－3, 轴号

IL　　: 轴 PID 调节积分限; 范围: 整型数(0-65535);　单位: 无。

返回码：

0 或-1, -1=不成功, 0=成功

§3.3.5　SetAxisOPC_9010

设置轴电压输出 0 点补偿

函数编号:112

函数原型：

C：　　short APIENTRY SetAxisOPC_9010(unsigned short Board_NO,unsigned short Axis_No,short int OPC_value);

解释：

设置轴电压输出 0 点漂移补偿值。每个当量大约为 0.0025V。

轴正在运行时是否可调用：是

参数：

Board_NO : 0－3, 板号
Axis_No　: 0－3, 轴号

OPC_value: 轴电压输出 0 点补偿; 整数型，　范围: -1000 - +1000　。

返回码：

0 或-1, -1=不成功, 0=成功

§3.3.6 SetAxisFVRate_9010

设置轴 PID 调节速度前馈系数

函数编号:103

函数原型：

C： short APIENTRY SetAxisFVRate_9010(unsigned short Board_NO,unsigned short Axis_No,unsigned long EcLine,double Speed_V,double ComV);

解释：

设置轴 PID 调节速度前馈计算系数。

轴正在运行时是否可调用：否

参数：

Board_NO ： 0－3, 板号
Axis_No ： 0－3, 轴号
EcLine ：轴电机编码器反馈线数 (四倍频前，即实际线数)
Speed_V ：命令电压对应的轴电机转速，当命令电压=10 时，就是轴电机的最高转速； 单位: RPM (转/分钟)。
ComV ： 命令电压，范围：0.1-10.0 伏；一般写 10。

返回码：

0 或-1, -1=不成功, 0=成功

§3.3.7 SetAxisFV_9010

设置轴 PID 调节速度前馈

函数编号:103

函数原型：

C： short APIENTRY SetAxisFV_9010(unsigned short Board_NO,unsigned short Axis_No,unsigned short FV);

解释：

设置轴 PID 调节速度前馈百分率

轴正在运行时是否可调用：是

参数：

Board_NO : 0－3, 板号
Axis_No : 0－3, 轴号

FV : 轴 PID 调节速度前馈系数; 范围: 整型数(0-65535); 单位: 无。
一般情况下，如果通过函数 SetAxisFVRate_9010 设置的 “轴电机编码器反馈线数”与“命令电压对应轴电机转速”准确无误，FV 可设为 100。

返回码：

0 或-1, -1=不成功, 0=成功

§3.3.8 SetAxisPEL_9010

设置轴位置闭环误差极限

函数编号:104

函数原型：

C: short APIENTRY SetAxisPEL_9010(unsigned short Board_NO,unsigned short Axis_No,unsigned short PosErrL);

解释：

设置轴位置闭环跟随误差极限，轴跟随误差超过该设定值时，9010 将产生报警，函数 GetErrorNo_9010 将返回 36（轴位置跟随误差超限），函数 ReadAxisState_9010 将返回-13 值。

轴正在运行时是否可调用：否

参数：

Board_NO : 0－3, 板号
Axis_No　: 0－3, 轴号

PosErrL　: 设置轴位置闭环误差极限; 范围: 整型数(0-65535);　单位: 无。

返回码：

0 或-1, -1=不成功, 0=成功

§ 3.3.9　SetEncoderX4X1_9010

设编码器倍频数　***（固件 5.2 以上版）***

函数编号:104

函数原型：

C：　short APIENTRY SetEncoderX4X1_9010(unsigned short Board_NO,unsigned short Axis_No,short X4X1);

解释：

该函数设置轴 ***位置反馈编码器*** 和 ***附加编码器*** 的内部倍频数，9010 卡初始化后缺省状态下为 4 倍频，特殊情况下可设 1 倍频。

对于闭环轴（工作模式为 2 或 3），初始化 9010 卡后，应首先设编码器倍频数（如果需要设 1 倍频），注意时机。

轴正在运行时是否可调用：否

参数：

Board_NO ：　0－3，　板号
Axis_No　：　0－4，　0-3 轴号；4=附加编码器

X4X1　　：0、1；0=4 倍频；1=1 倍频。

返回码：

0 或-1, -1=不成功, 0=成功

§3.4 轴状态读取函数

§3.4.1 ReadAxisTheoryPos_9010

读取轴的当前理论位置

函数编号:2

函数原型：

C: long APIENTRY ReadAxisTheoryPos_9010(unsigned short Board_NO,unsigned short Axis_No);

解释：

返回轴的当前实际理论位置。例如：轴从零位正向旋转了 100 个步距角，又负向旋转了 40 个步距角，则调用 ReadAxisPos_9010 函数将返回 60（轴的当前位置）。

注意: 在 9010 卡内部，每个轴有 2 个位置计数器：

1 轴理论位置计数器；（在 9010 内部对轴所发脉冲数进行计数）
2 轴编码器反馈位置计数器。

轴正在运行时是否可调用：是

参数：

Board_NO 卡号，可能值 0，1，2，3
Axis_No 轴号，可能值 0，1，2，3

返回码：

轴的位置; 范围，长整型数； 单位，脉冲个数。
当返回轴位置= -2147483648 时,表示不成功,有错误产生。

参考函数：

ReadAxisVel_9010

§3.4.2 ReadAxisEncodePos_9010

读取轴的编码器位置

函数编号:2

函数原型：

C: long APIENTRY ReadAxisEncodePos_9010 (unsigned short Board_NO,unsigned short Axis_No);

解释：

返回轴的当前编码器实际位置。

轴正在运行时是否可调用：是

参数：

Board_NO 卡号，可能值 0，1，2，3

Axis_No 轴号，可能值 0，1，2，3

返回码：

轴的位置; 范围，长整型数； 单位，脉冲个数。

当返回轴位置= -2147483648 时,表示不成功,有错误产生。

参考函数：

ReadAxisVel_9010

§3.4.3 ReadAxisPos_9010

返回轴当前位置

函数编号:2

函数原型：

C: long APIENTRY ReadAxisPos_9010(unsigned short Board_NO,unsigned short Axis_No);

解释：

返回轴的当前位置。根据轴的工作模式，该函数返回值不一样，总结如下:

0=轴无效;	返回 0 值
1=开环脉冲模式;	返回轴理论位置
2=位置闭环脉冲输出模式;	返回轴编码器反馈位置值
3=位置闭环模拟量输出模式;	返回轴编码器反馈位置值
4=单纯电压输出模式;	返回 0 值
5=单纯 PWM 脉冲输出模式;	返回 0 值

轴正在运行时是否可调用：是

参数：

Board_NO　卡号，可能值 0，1，2，3

Axis_No　轴号，可能值 0，1，2，3

返回码：

轴的位置; 范围，长整型数； 单位，脉冲个数。

当返回轴位置= -2147483648 时,表示不成功,有错误产生。

参考函数：

ReadAxisVel_9010

§3.4.4 ReadAxisTECV_9010

读取轴螺距误差补偿数据

函数编号:2

函数原型：

C: long APIENTRY ReadAxisTECV_9010 (unsigned short Board_NO,unsigned short Axis_No);

解释：

如果轴设了螺距误差补偿功能，该函数返回轴的当前轴螺距误差补偿实际值。

轴正在运行时是否可调用：是

参数：

Board_NO　卡号，可能值 0，1，2，3

Axis_No　轴号，可能值 0，1，2，3

返回码：

轴的位置; 范围，长整型数； 单位，脉冲个数。

参考函数：

§3.4.5 ReadAxisVel_9010

返回轴的当前速度

函数编号:26

函数原型：

C: long APIENTRY ReadAxisVel_9010(unsigned short Board_NO,unsigned short Axis_No);

解释：

返回轴的当前实际速度。

轴正在运行时是否可调用：是

参数：

Board_NO 卡号，可能值 0，1，2，3
Axis_No 轴号，可能值 0，1，2，3

返回码：

轴的当前速度； 范围，长整型数；单位，Hz，轴的脉冲频率。
当返回轴位置= -2147483648 时,表示不成功,有错误产生。

参考函数：

ReadAxisPos_9010

§3.4.6 ReadAxisState_9010

读取轴的状态

函数编号:27

函数原型：

C: short APIENTRY ReadAxisState_9010(unsigned short Board_NO,unsigned short Axis_No);

解释：

这个函数将返回轴的状态。

轴正在运行时是否可调用：是

参数：

Board_NO 卡号，可能值 0，1，2，3

Axis_No 轴号，可能值 0，1，2，3

返回码：

1: 轴在运动中

0: 轴在停止状态,轴位置到达,或被清零(Home_9010 命令)

-1: 轴 GoHome OK

-2: 轴在 GoHome 中暂时停止

-3: 轴被 StopAxis_9010 命令停止

-4: 轴被 AbortAxis_9010 命令停止

-5: 轴被 CeaseAxis_9010 命令停止

-6: 轴被 插补 命令停止

-7: 轴被正限位停止

-8: 轴被负限位停止

-9: 轴位置寄存器溢出被停止

-10: 轴被外部 IO 急停停止

-11: 轴在跟随模式下速度为 0

-12: 轴编码器位置寄存器溢出

-13: 轴跟随误差超限

255：读错误

注意：当轴有位置捕捉时（***轴位置捕获标志位=1*** 函数 IsPosCatch_9010 获得），函数 ReadAxisState_9010 只返回 2 种状态：

1: 轴在运动中

0: 轴在停止状态

§3.4.7 IsPosCatch_9010

读取轴位置捕获标志位 （固件 5.2 以上版）

函数编号:1021

函数原型：

C: short APIENTRY IsPosCatch_9010(unsigned short Board_NO,unsigned short Axis_No);

解释：

这个函数将返回轴是否有位置捕捉。

轴正在运行时是否可调用：是

参数：

Board_NO 卡号，可能值 0，1，2，3

Axis_No 轴号，可能值 0，1，2，3

返回码：

1,0 或-1,

1=有轴位置捕获,0=没有轴位置捕获

-1=不成功,0=成功

§3.4.8 ReadCatchPos_9010

读取轴的捕获位置

函数编号:2

函数原型：

C: long APIENTRY ReadCatchPos_9010(unsigned short Board_NO,unsigned short Axis_No,short ResetFlag);

解释：

返回轴的捕捉位置。根据轴的工作模式不同，捕捉位置将临时占据轴不同的***位置寄存器***（参看 ***§2.5.6 位置捕捉函数*** 节），总结如下:

0=轴无效; 返回 0 值

1=开环脉冲模式;	返回轴编码器反馈位置寄存器值
2=位置闭环脉冲输出模式;	返回轴理论位置寄存器值
3=位置闭环模拟量输出模式;	返回轴理论位置寄存器值
4=单纯电压输出模式;	返回轴理论位置寄存器值
5=单纯 PWM 脉冲输出模式;	返回轴理论位置寄存器值

轴正在运行时是否可调用：是

参数：

Board_NO ：卡号，可能值 0，1，2，3

Axis_No ：轴号，可能值 0，1，2，3

ResetFlag ： 0,1 0=轴位置捕获标志位保持; 1=轴位置捕获标志位复位

注意：

轴位置捕获标志位复位后，将解除 **捕捉位置** 临时占据的轴***位置寄存器***，这种解除不是立即有效，由于 9010 卡的工作原理（参看 ***§2.4 9010 卡工作原理*** 节），要延时 2 毫秒。

返回码：

轴的捕捉位置; 范围，长整型数； 单位，脉冲个数。

当返回轴位置= -2147483648 时,表示不成功,有错误产生。

参考函数：

ReadAxisTheoryPos_9010

ReadAxisEncodePos_9010

§3.4.9 ResetPosCatch_9010

轴位置捕获标志位复位 ***（固件 5.2 以上版）***

函数编号:124

函数原型：

C: long APIENTRY ResetPosCatch_9010(unsigned short Board_NO,unsigned short Axis_No);

解释：

轴位置捕获标志位复位。注意：轴位置捕获标志位复位后，将解除 **捕捉位置** 临时占据的轴***位置寄存器***，这种解除不是立即有效，由于 9010 卡的工作原理（参看 ***§2.4 9010 卡工作原理*** 节），要延时 2 毫秒。

轴正在运行时是否可调用：是

参数：

Board_NO　　：卡号，可能值 0，1，2，3

Axis_No　　：轴号，可能值 0，1，2，3

返回码：

0 或-1,-1=不成功,0=成功

参考函数：

IsPosCatch_9010

§3.5 轴螺距误差补偿设置函数

§3.5.1 SetAxisTEC_9010

设置轴螺距反向间隙补偿

函数编号:73

函数原型：

C:　　short APIENTRY SetAxisTEC_9010(unsigned short Board_NO,unsigned short Axis_No,long ErrorV,short TimeNum);

解释：

该函数将设轴单纯的螺距反向间隙补偿值。一般原则是当轴回零点完成后（Home_9010 或 GoHome_9010 命令之后），调用该函数来设轴的反向间隙补偿值。在调用该函数那一刻，9010 卡自动判断轴方向，当轴运动方向与该时刻方向相反时，9010 卡将开始进行螺距反向间隙补偿。注意：Home_9010 命令或 GoHome_9010 命令自动取消轴螺距反向间隙补偿。

轴正在运行时是否可调用：否

参数：

Board_NO　：卡号，可能值 0，1，2，3

Axis_No　：轴号，可能值 0，1，2，3

ErrorV　：轴螺距反向间隙补偿值；范围: 0-32767；　单位：脉冲个数。

TimeNum　：补偿所用时间；范围: 1-20；　单位：毫秒。一般原则是反向间隙补偿值越大，所用时间应该越长。

返回码：

0 或-1,　　-1=不成功,0=成功

参考函数：

§3.5.2　SetAxisTECData_9010

设置轴螺距误差补偿数据

函数编号:111

函数原型：

C:　　short APIENTRY SetAxisTECData_9010(unsigned short Board_NO,unsigned short Axis_No,short Mode, short EffectNum,long BasePoint,long NodeLen,short Direc,short ReverseGap,short *TEData);

解释：

该函数将设轴螺距误差补偿模式和数据值。该函数要比 SetAxisTEC_9010 函数高级和复杂，一般原则是：

简单非精密应用场合，可用 SetAxisTEC_9010 函数进行单纯的螺距反向间隙补偿。

精密应用场合，且轴工作模式为 3（位置闭环模拟量输出模式，**必要条件**）时，可用该函数进行轴螺距误差补偿。

调用该函数后，SetAxisTEC_9010 函数自然失效。

9010 卡具体支持的补偿类型可分为三类：

① 单纯反向间隙补偿（用 SetAxisTEC_9010 函数实现）

② 反向间隙补偿＋单向螺距误差补偿

此种补偿类型中：有 1 组螺补数据，该组螺补数据数最大为 511 个。

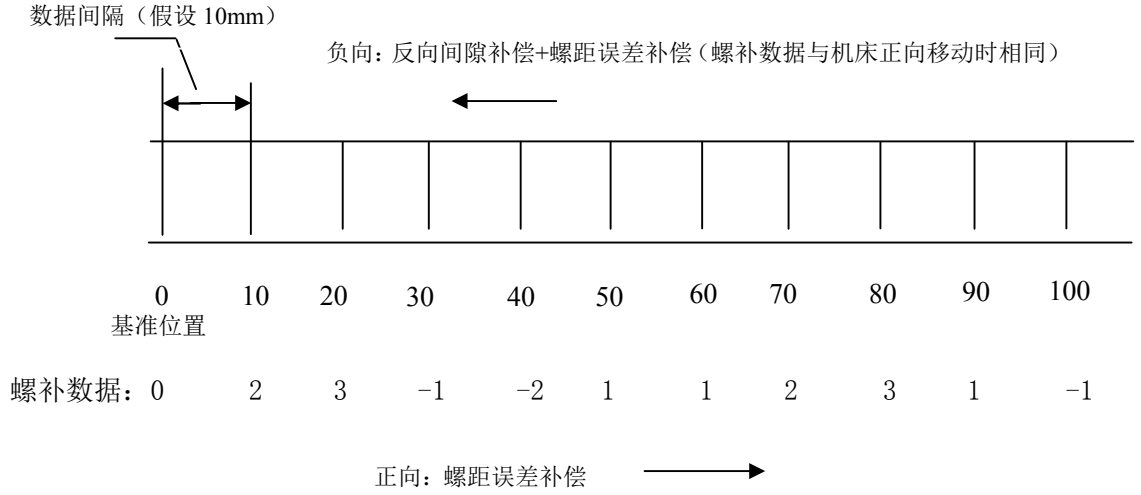
补偿方向：补偿方向选择 1 时，表示该轴正向移动时进行螺距误差补偿，该轴负向移动时，在螺距误差补偿数据的基础上再加上反向间隙补偿数据，进行螺距误差补偿。

补偿方向选择 -1 时，表示该轴负向移动时进行螺距误差补偿，该轴正向移动时，在螺距误差补偿数据的基础上再加上反向间隙补偿数据，进行螺距误差补偿。

基准位置：在测量滚珠丝杠螺距误差时的基准位置，可设丝杠上的任一点设为基点位置。

有效数据数：指一组螺补数据所包含的数据成员个数。

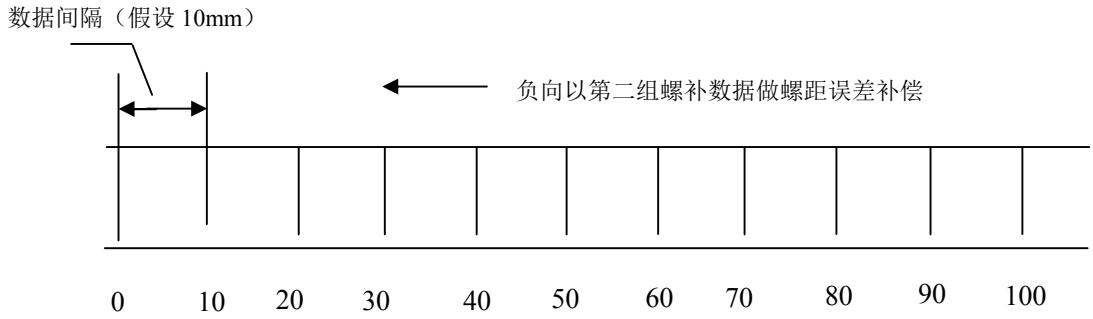
数据间隔： 测量一组螺补数据时，每个数据的间隔距离。



③ 双向螺距误差补偿

此种补偿类型中：有 2 组螺补数据，每组螺补数据最大为 255 个

补偿方向：补偿方向选择 1 时，表示在机床正向移动时以第一组数据做螺距误差补偿，负向移动时以第二组数据做螺距误差补偿；补偿方向为 -1 时，表示在机床负向移动时以第一组数据做螺距误差补偿，正向移动时以第二组数据做螺距误差补偿。



基准位置

正向以第一组螺补数据做螺距误差补偿 ⟶

轴正在运行时是否可调用：否

参数：

Board_NO : 0－3, 板号

Axis_No : 0－3, 轴号

Mode : 2 或 3 2=反向间隙+螺距误差补偿;3=双向螺距误差补偿

EffectNum : Mode=2 时，1-512； Mode=3 时，0-256； 有效数据数

BasePoint : 基点位置， 数组 TEData[0]对应的位置,单位：脉冲个数。

NodeLen : 数据间距离，单位：脉冲个数。

Direc :

1 或 -1；方向：Mode=2 时， 反向间隙补偿方向；Mode=3 时， 前(双向)一组数据方向

ReverseGap : 反向间隙补偿数据，范围：0-32767； 单位：脉冲个数。

TEData :

螺距误差补偿数据数组，数组个数固定为 512 个，每个数据范围：-32768 - +32767； 单位：脉冲个数。

如果补偿模式是 3=双向螺距误差补偿 时，第 1 组数据占据 0-255 数组；第 2 组数据占据 256-511 数组；

返回码：

0 或-1, -1=不成功,0=成功

参考函数：

§3.5.3 SetAxisTECWork_9010

启动/停止 轴螺距误差补偿

函数编号:111

函数原型：

C: short APIENTRY SetAxisTECWork_9010(unsigned short Board_NO,unsigned short Axis_No,short Work);

解释：

当用函数 SetAxisTECData_9010 向 9010 卡中传入螺补数据数据后，用该函数启动/停止螺距误差补偿。一般原则是当轴回零点完成后（GoHome_9010 命令之后），调用该函数来启动轴螺距误差补偿功能。

轴正在运行时是否可调用：否

参数：

Board_NO : 0－3, 板号

Axis_No : 0－3, 轴号

Work : 0 或 1 0=停止螺距误差补偿;1=启动螺距误差补偿

返回码：

0 或-1、-2： -1=不成功,-2=轴螺距误差补偿数据失效，0=成功

参考函数：

§3.6 单轴运动控制函数

§3.6.1 SetAxisPos _9010

设置梯形包络线的终点位置

函数编号:4

函数原型：

C: short APIENTRY SetAxisPos_9010(unsigned short Board_NO,unsigned short Axis_No,long position);

解释：

设置轴的终点位置参数，在调用该函数后，用户再调用 startAxis_9010 函数，轴终点位置的新参数才有起作用。

轴正在运行时是否可调用：是

参数：

Board_NO	卡号，可能值 0，1，2，3
Axis_No	轴号，可能值 0，1，2，3
position	轴的位置; 范围: 长整型数; 单位: 脉冲个数。

返回码：

(0) 成功

(-1) 非法参数

参考函数：

SetAxisVel_9010

SetAxisAcc_9010

StartAxis_9010

§3.6.2 SetAxisVel_9010

设置梯形包络线的最大速度

函数编号:5

函数原型：

C: short APIENTRY SetAxisVel_9010(unsigned short Board_NO,unsigned short Axis_No,long velocity);

解释：

设置轴的最大速度，在调用该函数后，用户再调用 StartAxis_9010 函数，新参数才起作用。

轴正在运行时是否可调用：是

参数：

Board_NO	卡号，可能值 0，1，2，3
Axis_No	轴号，可能值 0，1，2，3
Velocity	轴的速度， 范围: 长整型数; 单位: Hz，轴的脉冲频率

返回码：

(0) 成功

(-1)　非法参数

参考函数：

SetAxisPos_9010

SetAxisAcc_9010

StartAxis_9010

§3.6.3　SetAxisStartVel _9010

设置轴的起跳速度

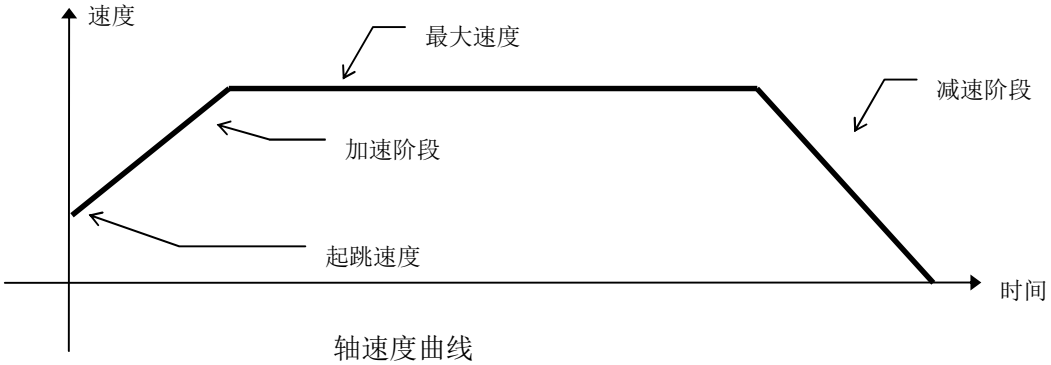
函数编号:88

函数原型：

C:　short APIENTRY SetAxisStartVel_9010(unsigned short Board_NO,short Axis_No,long velocity);

解释：

设置轴的起跳速度。在 9010 初始化时,每个轴的系统缺省起跳速度为 0，轴的起跳速度只对本轴单独运行函数有影响，对插补函数没有影响。



轴速度曲线

轴正在运行时是否可调用：否

参数：

Board_NO　卡号，可能值 0，1，2，3

Axis_No　轴号，可能值 0，1，2，3

velocity:　轴的起跳速度，可正负，单位：脉冲数 / 秒（Hz）。

返回码：

(0)　成功；　(-1)　不成功

参考函数：

§3.6.4　SetAxisStopVel _9010

设轴的停止速度

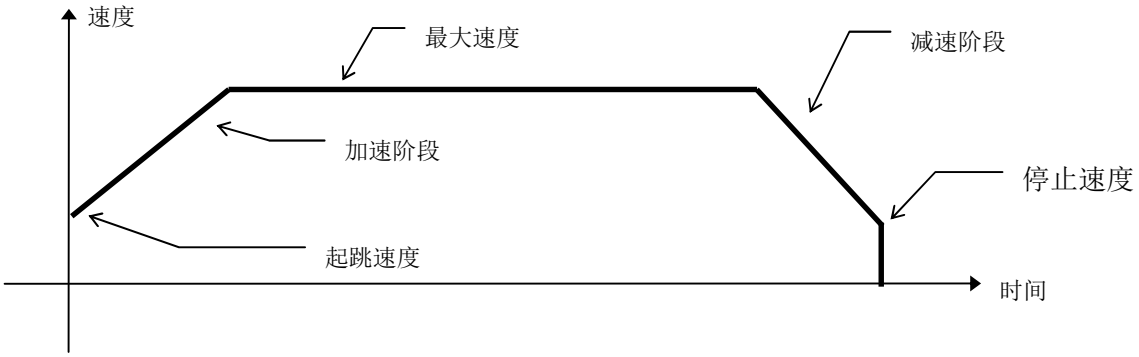
函数编号:95

函数原型：

C:　short APIENTRY SetAxisStopVel_9010(unsigned short Board_NO,short Axis_No,long velocity);

解释：

设置轴的停止速度。在 9010 初始化时,每个轴的系统缺省停止速度为 16，轴的停止速度只对本轴单独运行函数有影响，对插补函数没有影响。



轴速度曲线

轴正在运行时是否可调用：否

参数：

Board_NO　卡号，可能值 0，1，2，3

Axis_No　轴号，可能值 0，1，2，3

velocity : 轴的停止速度; 范围: 正长整型数;　单位: Hz，轴的脉冲频率。缺省值: 16

返回码：

(0)　成功；　(-1)　不成功

参考函数：

§3.6.5 SetAxisDec_9010

设轴的减速度,固件 3.0 版

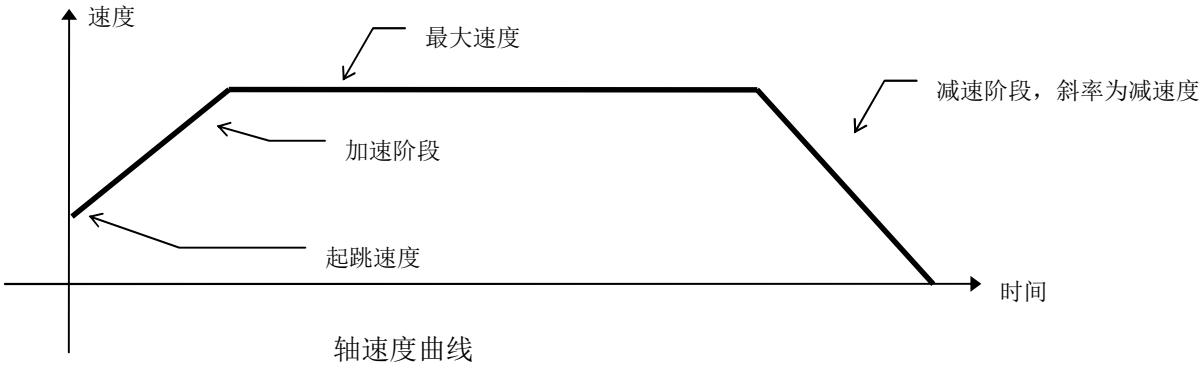
函数编号:96

函数原型：

C: short APIENTRY SetAxisDec_9010(unsigned short Board_NO,short Axis_No,long deceleration);

解释：

设置轴的减速度值。在 9010 初始化时,每个轴的系统缺省起跳速度为 1000，轴的减速度值只对本轴单独运行函数有影响，对插补函数没有影响。



轴速度曲线

轴正在运行时是否可调用：否

参数：

Board_NO 卡号，可能值 0，1，2，3

Axis_No 轴号，可能值 0，1，2，3

deceleration: 轴的减速度; 范围: 长整型数,必须是正值; 单位: 脉冲数 / 秒平方。

返回码：

(0) 成功； (-1) 不成功

参考函数：

§3.6.6 SetAxisAcc_9010

设置梯形包络线的加速度，绝对值

函数编号:6

函数原型：

C： short APIENTRY SetAxisAcc_9010(unsigned short Board_NO,unsigned short Axis_No,long acceleration);

解释：

设置轴的加速度参数。在控制卡初始化时，每个轴的加速度均为 0。

轴正在运行时是否可调用：否

参数：

Board_NO	卡号，可能值 0，1，2，3
Axis_No	轴号，可能值 0，1，2，3
Acceleration	轴的加速度; 范围: 长整型数,必须是正值; 单位: 脉冲数 / 秒平方。

返回码：

(0) 成功

(-1) 非法参数

参考函数：

SetAxisPos_9010

SetAxisVel_9010

StartAxis_9010

SetAxisSAcce_9010

§3.6.7 SetAxisSAcc _9010

设置设置轴 S 型加速度

函数编号:72

函数原型：

C： short APIENTRY SetAxisSAcce_9010(unsigned short Board_NO,unsigned short Axis_No,short PowerFlag);

解释：

（保留函数）

轴正在运行时是否可调用：否

参数：

Board_NO ： 卡号，可能值 0，1，2，3

Axis_No ： 轴号，可能值 0，1，2，3

PowerFlag ： 1-4 S 型加速度(1,2,3,4)指数，1=直线形加速度。系统缺省值：2

返回码：

(0) 成功

(-1) 非法参数

参考函数：

SetAxisAcc_9010

§3.6.8 StartAxis_9010

更新梯形包络线参数，轴开始运动

函数编号:7

函数原型：

C: short APIENTRY StartAxis_9010(unsigned short Board_NO,unsigned short Axis_No);

解释：

轴以位置模式开始运行，在这之前，必须设好轴的位置、速度和加速度。

轴正在运行时是否可调用：是

参数：

Board_NO 卡号，可能值 0，1，2，3

Axis_No 轴号，可能值 0，1，2，3

返回码：

(0) 成功

(-1)　不成功

参考函数：

SetAxisPos_9010

SetAxisVel_9010

SetAxisAcc_9010

§3.6.9　StartAxisVel_9010

设置轴为速度模式

函数编号:8

函数原型：

C:　short APIENTRY StartAxisVel_9010(unsigned short Board_NO,unsigned short Axis_No,long velocity);

解释：

轴以速度模式开始运行，在这之前，必须设好轴的加速度。

轴正在运行时是否可调用：是

参数：

Board_NO　卡号，可能值 0，1，2，3

Axis_No　轴号，可能值 0，1，2，3

velocity:　轴的最大速度，可正负，单位：脉冲数 / 秒（Hz）。

返回码：

(0)　成功

(-1)　不成功

§3.6.10　AbortAxis_9010

轴停止,在 20 毫秒之内减速停止

函数编号:19

函数原型：

C: short APIENTRY AbortAxis_9010(unsigned short Board_NO,unsigned short Axis_No);

解释：

这个函数将使轴的速度在 10 毫秒之内降为零。

轴正在运行时是否可调用：是

参数：

Board_NO 卡号，可能值 0，1，2，3
Axis_No 轴号，可能值 0，1，2，3

返回码：

(0) 成功
(-1) 不成功

参考函数：

StopAxis_9010
CeaseAxis_9010

§3.6.11 SetAxisStopDec_9010

设轴的 Stop 命令的减速度 （固件 V5.1 以上版）

函数编号:122

函数原型：

C: short APIENTRY SetAxisStopDec_9010(unsigned short Board_NO,short Axis_No,long deceleration);

解释：

这个函数设 StopAxis_9010 函数的减速度值。系统缺省值为 1000。

轴正在运行时是否可调用：是

参数：

Board_NO　：卡号，可能值 0，1，2，3

Axis_No　：轴号，可能值 0，1，2，3

deceleration　：　轴的减速度; 范围: 长整型数,必须是正值;　单位: 脉冲数 / 秒平方。

返回码：

(0)　成功

(-1)　不成功

参考函数：

StopAxis_9010

§3.6.12　StopAxis_9010

轴停止, 按所设加速度减速停止

函数编号:9

函数原型：

C:　short APIENTRY StopAxis_9010(unsigned short Board_NO,unsigned short Axis_No);

解释：

这个函数将使轴的速度以用户设定的减速度方式降为零。

轴正在运行时是否可调用：是

参数：

Board_NO　卡号，可能值 0，1，2，3

Axis_No　轴号，可能值 0，1，2，3

返回码：

(0)　成功

(-1)　不成功

参考函数：

AbortAxis_9010

CeaseAxis_9010

§3.6.13 CeaseAxis_9010

轴停止,立即停止,无减速过程

函数编号:20

函数原型:

C: short APIENTRY CeaseAxis_9010(unsigned short Board_NO,unsigned short Axis_No);

解释:

这个函数将使轴的速度立即降为零。

轴正在运行时是否可调用:是

参数:

Board_NO 卡号,可能值 0,1,2,3

Axis_No 轴号,可能值 0,1,2,3

返回码:

(0) 成功

(-1) 不成功

参考函数:

StopAxis_9010

AbortAxis_9010

§3.6.14 Home_9010

轴位置寄存器清零

函数编号:3

函数原型:

C: short APIENTRY Home_9010(unsigned short Board_NO,unsigned short Axis_No);

解释：

这个函数将把轴内部位置计数器清零。

注意: 在 9010 卡内部，每个轴有 2 个位置计数器：

1 轴理论位置计数器；（在 9010 内部对轴所发脉冲数进行计数）
2 轴编码器反馈位置计数器。

当轴的工作模式不同时，该函数作用有所不同，总结如下：

1=开环脉冲模式　　：轴理论位置计数器清零
2=位置闭环脉冲输出模式　　：轴理论位置计数器 和 轴编码器反馈位置计数器 清零
3=位置闭环模拟量输出模式　　：同 2

轴正在运行时是否可调用：否

参数：

Board_NO　　卡号，可能值 0，1，2，3
Axis_No　　轴号，可能值 0，1，2，3

返回码：

(0)　成功
(-1)　不成功

参考函数：

ReadAxisPos_9010

§3.6.15　HomeFB_9010

轴位置编码器清零

函数编号:108

函数原型：

C:　　short APIENTRY HomeFB_9010(unsigned short Board_NO,unsigned short Axis_No);

解释：

这个函数将把轴编码器反馈位置计数器清零。

当轴的工作模式是 2（位置闭环脉冲输出模式）和 3（位置闭环模拟量输出模式）时，应绝对禁止用该函数对轴编码器反馈位置计数器清零。设该函数的目的是为了当轴工作在 0、1、4、5 模式下，可单独使用该轴的 编码器反馈位置计数器。

轴正在运行时是否可调用：工作模式 0、1、4、5 时， 是

参数：

Board_NO　卡号，可能值 0，1，2，3
Axis_No　轴号，可能值 0，1，2，3

返回码：

(0)　成功
(-1)　不成功

参考函数：
ReadAxisPos_9010

§3.6.16　GoHome_9010

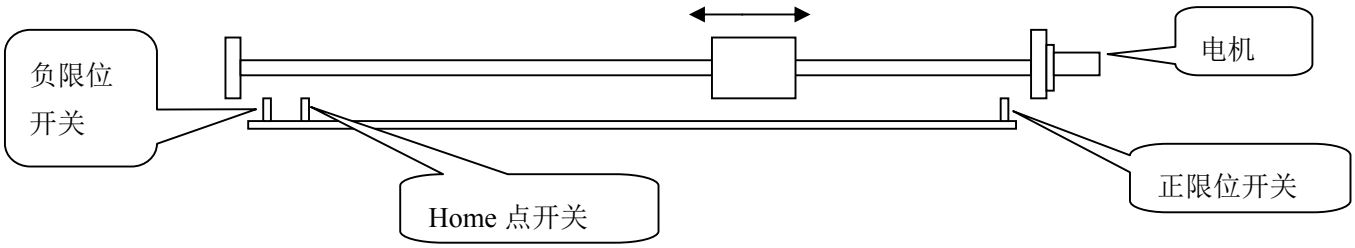
轴回 Home 点
函数编号:24
函数原型：

C:　short APIENTRY GoHome_9010(unsigned short Board_NO,unsigned short Axis_No,long goHomeVel,long LeaveHomeVel,long LeaveHomePos,long LookZIndexVel,long PulseNum,short Z_IndexFlag);

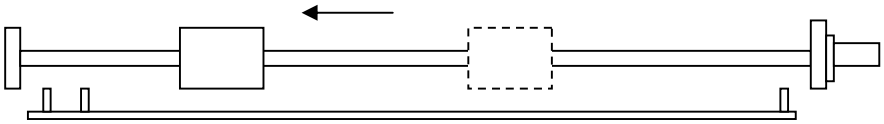
解释：

这个函数将使轴进行真实回 Home 点操作。回 Home 点过程示意如下：

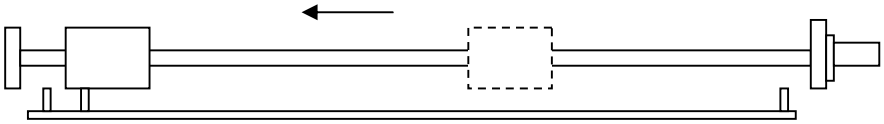
连线图示意：



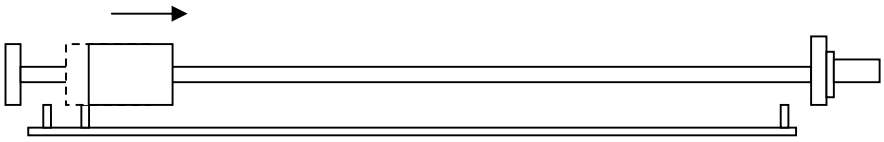
步骤 1：从当前位置向 Home 点移动



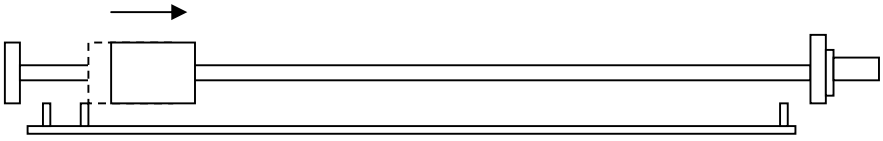
步骤 2：压上 Home 点



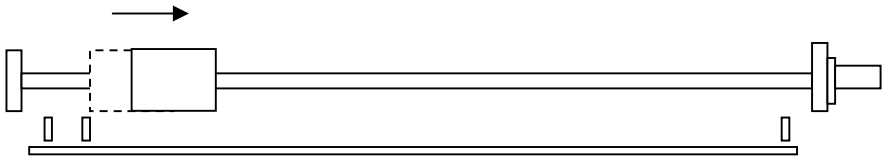
步骤 3：脱离 Home 点



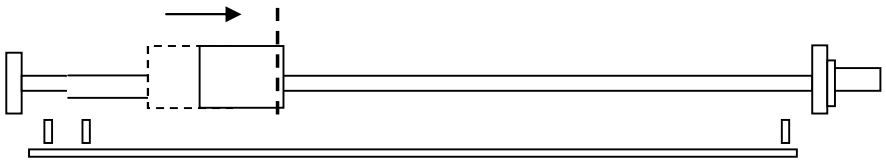
步骤 4：脱离 Home 点 一段距离



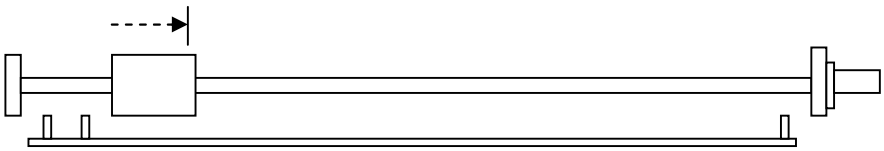
步骤 5：启动 Index 捕获，继续运动，在一圈内找 Index 点，如果不找 Index 点，则转步骤 7。



步骤 6：当 Index 信号产生时，转步骤 7。



步骤 7：停止运动，轴位置计数器清零，GoHome 完成



在回 Home 点时，回 Home 点方向、速度，脱离 Home 点方向、速度及距离均可由参数设定。只有在轴 Home 点有效时（由函数 SetAxisIO_9010 设定），GoHome 函数才有效。

注意：在 GoHome_9010 命令之前，确保有 SetAxisAcc_9010 （设置轴加速度）命令

轴正在运行时是否可调用：否

参数：

Board_NO : 0－3, 板号

Axis_No : 0－3, 轴号

goHomeVel : 轴回 Home 点速度; 范围: 长整型数; 可正负； 单位: Hz，轴的脉冲频率。
决定步骤 1 的方向和速度。

LeaveHomeVel: 轴离开 Home 点速度; 范围: 长整型数; 可正负；单位: Hz，轴的脉冲频率。
决定步骤 3、4 的方向和速度。该速度值一般要选择小一点。

LeaveHomePos: 轴离开 Home 点距离; 范围: 长整型数,必须是正值； 单位: 脉冲个数。
决定步骤 4 的距离。

LookZIndexVel: 轴找一转脉冲速度; 范围: 长整型数; 单位: Hz，轴的脉冲频率。

PulseNum : 轴编码器反馈每转脉冲数(或一转脉冲的间隔脉冲个数 (光栅尺反馈))

Z_IndexFlag : 0 或 1;当轴的工作模式为 2 或 3 时,可设回零找一转脉冲模式:
0=回零不找一转脉冲;1=回零找一转脉冲

返回码：

(0) 成功

(-1) 非法参数

参考函数：

ReadAxisPos_9010、Home_9010、SetAxisIO_9010

§3.6.17 LookZIndex_9010

轴找一转脉冲(轴回 Home 点的分解动作)

函数编号:114

函数原型：

C: short APIENTRY LookZIndex_9010(unsigned short Board_NO,unsigned short Axis_No,long LookZIndexVel,long PulseNum);

解释：

这个函数将使轴单纯找一转脉冲（Index 点）操作。也是轴回 Home 点的分解动作（步骤 5 以后动作）

轴正在运行时是否可调用：否

参数：

Board_NO : 0－3, 板号

Axis_No : 0－3, 轴号

LookZIndexVel: 轴找一转脉冲速度; 范围: 长整型数; 可正负； 单位: Hz，轴的脉冲频率。

PulseNum : 轴编码器反馈每转脉冲数(或一转脉冲的间隔脉冲个数 (光栅尺反馈))

返回码：

(0) 成功

(-1) 非法参数

参考函数：

ReadAxisPos_9010、Home_9010、SetAxisIO_9010

§3.6.18 SetAxisOffset_9010

设置轴位置偏移值

函数编号:84

函数原型：

C: short APIENTRY SetAxisOffset_9010(unsigned short Board_NO,unsigned short Axis_No,long Offset);

解释：

该函数将预设轴位置计数器的值。Home_9010 命令或 GoHome_9010 命令将轴位置的偏移值（预设轴位置寄存器的值）清零。

注意: 在 9010 卡内部，每个轴有 2 个位置计数器：1 理论位置计数器；2 轴编码器反馈位置计数器。

当轴的工作模式不同时，该函数作用有所不同，总结如下：

1=开环脉冲模式　　：设轴理论位置计数器初值
2=位置闭环脉冲输出模式　　：设轴 理论位置计数器 和 轴编码器反馈位置计数器 初值
3=位置闭环模拟量输出模式　　：同 2

轴正在运行时是否可调用：否

参数：

Board_NO　　: 卡号，可能值 0，1，2，3
Axis_No　　: 轴号，可能值 0，1，2，3
offset　　: 轴位置的偏移值; 范围: 长整型数;　单位: 脉冲个数。

返回码：

0 或-1,　　-1=不成功,0=成功

参考函数：

§3.6.19　SetAxiFBOffset_9010

设轴位置编码器偏移值

函数编号:109

函数原型：

C:　　short APIENTRY SetAxisFBOffset_9010(unsigned short Board_NO,unsigned short Axis_No,long Offset);

解释：

该函数将预设 轴轴编码器反馈位置计数器 的初值。

当轴的工作模式是 2（位置闭环脉冲输出模式）和 3（位置闭环模拟量输出模式）时，应 绝对禁止用该函数对轴编码器反馈位置计数器设初值。设该函数的目的是为了当轴工作在 0、1、4、5 模式下，可单独使用该轴的 编码器反馈位置计数器。

轴正在运行时是否可调用：工作模式 0、1、4、5 时， 是

轴正在运行时是否可调用：否

参数：

Board_NO ：卡号，可能值 0，1，2，3

Axis_No ：轴号，可能值 0，1，2，3

offset ：轴轴编码器反馈位置计数器的偏移值; 范围: 长整型数; 单位: 脉冲个数。

返回码：

0 或-1, -1=不成功,0=成功

参考函数：

§3.7 电子齿轮与位置跟随

§3.7.1 StartAxisEGear_9010

设置轴电子齿轮运动

函数编号:75

函数原型：

C: short APIENTRY SetAxisEGear_9010(unsigned short Board_NO,unsigned short Axis_No,unsigned short F_Axis_No,double Rate,double Kp);

解释：

该函数设定轴以电子齿轮模式工作，设定完后立即生效。主从关系、齿轮比可通过参数

设定。主从关系可随意设定，只要不违反自己随自己或父子孙的伦理关系就可以。设定时要求主动轴、从动轴均在停止状态(9010 卡会在两轴停止状态那一刻读取两轴的当前位置,以此为两轴的开始位置)，在设定完后从动轴为电子齿轮模式，而主动轮可以为其它任意模式(除了 Home 和 GoHome 命令)。

轴正在运行时是否可调用：否

参数：

Board_NO ：卡号，可能值 0，1，2，3

Axis_No ：0－3, 从动轴号

F_Axis_No ：0－3, 主动轴号

Rate ： 从动轴跟随比率：

设：

A1=从动轴开始位置，A2=从动轴实时位置，单位：脉冲数

B1=主动轴开始位置，B2=主动轴实时位置，单位：脉冲数

则：

$$Rate = \frac{A2 - A1}{B2 - B1}$$

可正负; 范围: 绝对值 0.001-1000; 单位: 无; 分辨率: 0.001

Kp ： 跟随 PID 调节系数; 范围: 0.001-1000; 单位: 无; 分辨率: 0.001，一般设为 0.1、0.2 就可。

返回码：

(0) 成功

(-1) 不成功

参考函数：

CancelAxisFEG_9010

§3.7.2 SetActiveEncoder_9010

在轴跟随编码器运动之前,设主动编码器号

函数编号:121

函数原型：

C: short APIENTRY SetActiveEncoder_9010(unsigned short Board_NO,unsigned short Axis_No);

解释：

该函数为函数 SetAxisFE_9010 服务，指示主动编码器是哪个。

在 9010 卡中有 5 个编码器计数器（4 个轴每轴 1 个，还有一个附加编码器），当设定轴跟随编码器运动时，这 5 个编码器计数器都可能是主动编码器，限制条件是当轴工作在 2、3 模式下，不能自己跟随自己的编码器。在 9010 卡中每一时刻主动编码器只有一个，9010 卡初始化时，缺省的主动编码器是附加编码器。

轴正在运行时是否可调用：根据参数决定

参数：

Board_NO ：卡号，可能值 0，1，2，3

Axis_No ： 0－4, 0-3 对应轴号,4=附加编码器,缺省为 4

返回码：

(0) 成功

(-1) 不成功

参考函数：

CancelAxisFEG_9010

§3.7.3 SetAxisFE_9010

设置轴跟随编码器运动

函数编号:74

函数原型：

C: short APIENTRY SetAxisFE_9010(unsigned short Board_NO,unsigned short Axis_No,double Rate,double Kp,short Mode1,short Mode2,short Mode3);

解释：

该函数设定轴以编码器控制模式工作，设定完后立即生效。跟随关系、跟随比可通过参数设定。在设定完后轴为编码器控制模式工作，这时编码器(或接手摇轮、或是某一轴的位置反馈)的运动将带动该轴运动，这当中编码器不能执行编码器位置清零命令(HomeEncode_9010 命令)。

轴正在运行时是否可调用：根据参数决定

参数：

Board_NO ：卡号，可能值 0，1，2，3

Axis_No ：0－3, 轴号

Rate ： 轴跟随比率：

设：

A1=从动轴开始位置，A2=从动轴实时位置，单位：脉冲数

B1=编码器开始位置，B2=编码器实时位置，单位：脉冲数

则：

$$Rate = \frac{A2 - A1}{B2 - B1}$$

可正负; 范围: 绝对值 0.001-1000; 单位: 无; 分辨率: 0.001

Kp ： 跟随 PID 调节系数; 范围: 0.001-1000; 单位: 无; 分辨率: 0.001，一般设为 0.1、0.2 就可。

Mode1 ： 0=速度模式,1=位置模式

在速度模式下，轴的位置跟随不会严格按跟随比率来执行，运动效果更为柔和，而位置模式则会严格按跟随比率执行，运动柔和与否可用参数 Kp 调节。

Mode2 ： 0= 不自动清零， 1=自动清零

9010 卡的编码器计数器虽为 32 位，但为了防止编码器计数器溢出，可在开始时自动将编码器计数器清零。

当主动编码器是某轴的位置反馈编码器时，且该轴工作在 2、3 模式时，应绝对禁止用自动清零模式

Mode3 ： 0=停止状态跟随，1=可运动状态跟随

当轴的速度不为零时（运动状态）可立即转为编码器跟随模式。在实践中，比如数控铣床的攻丝、数控车床的切螺纹等操作，均是某一轴在运动状态下过渡到与编码器跟随。

返回码：

(0) 成功

(-1) 不成功

参考函数：

CancelAxisFEG_9010

§3.7.4 CancelAxisFEG_9010

取消轴跟随运动

函数编号:76

函数原型：

C: short APIENTRY CancelAxisFEG_9010(unsigned short Board_NO,unsigned short Axis_No);

解释：

当轴以电子齿轮模式工作或以编码器控制模式工作时，要取消这种模式，则可用该函数解除轴的这两种工作模式。解除轴跟随模式只能用该函数或轴停止命令（StopAxis_9010、AbortAxis_9010、CeaseAxis_9010）。 (插补模式下的编码器控制模式除外,看函数LM_LineFE_9010)

轴正在运行时是否可调用：否

参数：

Board_NO ：卡号，可能值 0，1，2，3

Axis_No ：0－3, 轴号

返回码：(0) 成功; (-1) 不成功

§3.7.5 GetEnAuto0Flag_9010

获得主动编码器自动清零标志

函数编号:80

函数原型：

C: short APIENTRY GetEnAuto0Flag_9010(unsigned short Board_NO);

解释：

该函数将获得主动编码器位置计数器是否被 9010 卡自动清零。在执行 LM_LineFE_9010 命令时，为了防止主动编码器计数器溢出，9010 卡可能对主动编码器计数器清零。

轴正在运行时是否可调用：是

参数：

卡号，可能值 0，1，2，3

返回码：

0:

1:　　主动编码器计数器曾经被 9010 卡自动清零过

255:　　命令不成功

参考函数：

ReadEncoderPos_9010

LM_LineFE_9010

ResetEn0Flag_9010

§3.7.6　ResetEn0Flag_9010

主动编码器自动清零标志置 0

函数编号:79

函数原型：

C:　　short APIENTRY ResetEn0Flag_9010(unsigned short Board_NO);

解释：

当用函数 GetEnAuto0Flag_9010 得到主动编码器位置计数器被自动清零（编码器自动清零标志位在 9010 卡中被置 1），用该函数将主动编码器自动清零标志位置 0。

轴正在运行时是否可调用：是

参数：

卡号，可能值 0，1，2，3

返回码：

0 或-1，　-1=不成功, 0=成功

参考函数：

ReadEncoderPos_9010

LM_LineFE_9010

GetEnAuto0Flag_9010

§3.8 轴其它控制函数

§3.8.1 SetAxisDAOut_9010

设置轴输出电压

函数编号:105

函数原型:

C: short APIENTRY SetAxisDAOut_9010(unsigned short Board_NO,unsigned short Axis_No,double DA_Avlue);

VB:

Delphi:

解释:

设置轴输出电压。该函数只有在轴工作模式为:

4=单纯电压输出模式;

时有效。可以用 SetAxisOPC_9010 函数修正 0 点电压值。

参数:

Board_NO : 0—3, 板号

Axis_No : 0—3, 轴号

DA_Avlue : -10 - +10 V; 精度: 1/8000;

返回码:

0 或-1, -1=不成功, 0=成功

§3.8.2 SetAxisPWMOut_9010

设置轴 PWM 脉冲输出

函数编号:106

函数原型：

C： short APIENTRY SetAxisPWMOut_9010(unsigned short Board_NO,unsigned short Axis_No,long frequency,float Pulse_Highf);

VB：

Delphi:

解释：

设置轴 PWM 脉冲输出。该函数只有在轴工作模式为:

5=单纯 PWM 脉冲输出模式;

时有效。

参数：

Board_NO : 0－3, 板号
Axis_No : 0－3, 轴号

frequency: 1-1000000; PWM 输出频率; 单位: 赫兹(Hz)
Pulse_Highf: 占空比,范围:0.0-1.0; 精度:不小于 1%。

返回码：

0 或-1, -1=不成功, 0=成功

§3.8.3 AxisPWMStop_9010

轴 PWM 频率输出停止

函数编号:107

函数原型：

C： short APIENTRY AxisPWMStop_9010(unsigned short Board_NO,unsigned short

Axis_No);

解释：

轴 PWM 频率输出停止。该函数只有在轴工作模式为:

5=单纯 PWM 脉冲输出模式;

时有效。

在插补引擎正在运行当中是否可调用：

参数：

Board_NO : 0－3, 板号

Axis_No : 0－3, 轴号

返回码：

0 或-1, -1=不成功, 0=成功

§3.8.4 SetAxisMotorOnOff_9010

设轴电机 On 或 Off，使能

函数编号:110

函数原型：

C： short APIENTRY SetAxisMotorOnOff_9010(unsigned short Board_NO,unsigned short Axis_No,short OnOff);

解释：

该函数控制轴的伺服电机使能信号， 电机使能信号参看 ***§1.5 电机控制信号*** 节。

参数：

Board_NO : 0－3, 板号

Axis_No : 0－3, 轴号

OnOff : 0,1; 0=Off,1=On。

返回码：

0 或-1, -1=不成功, 0=成功

§3.9 通用 IO 读写函数

§3.9.1 Set_Emergency_Stop_9010

设置 IO 急停信号

函数编号:29

函数原型：

C: short APIENTRY Set_Emergency_Stop_9010(unsigned short Board_NO,unsigned short Mask,short Mode);

解释：

该函数设置 I1-I8 中的某一个输入点为急停信号输入点。9010 卡在初始化后，急停信号输入点是无效的。

轴正在运行时是否可调用：是

参数：

Board_NO : 0－3, 板号

Mask : 0、1－8, 急停信号与通用输入点 I1-I8 相联,0 表示无效,9010 卡上电时 缺省为无效。

注意：输入点必须是 1-8 点。

Mode : 1、2、3；

1=CeaseAxis_9010 模式轴停止；

2=AbortAxis_9010 模式轴停止

3=StopAxis_9010 模式轴停止

返回码：

(0) 成功 ; (-1) 不成功

§3.9.2 ReadIO_9010

读板上数字 IO（输入）点信号状态

函数编号:14

函数原型：

C: long APIENTRY ReadIO_9010(unsigned short Board_NO);

解释：

读取通用输入点 I1-I20 状态,返回数值转换为二进制时的 0-19 位有效。

轴正在运行时是否可调用：是

参数：

Board_NO : 0－3, 板号

返回码：

(>=0) 成功，0-19 位为通用输入点 I1-I20 状态

(-1) 不成功

参考函数：

ReadIOBit_9010

WriteIo_9010

§3.9.3 ReadIOBit_9010

按位读板上数字 IO（输入）点信号状态

函数编号:14

函数原型：

C: short APIENTRY ReadIOBit_9010(unsigned short Board_NO,short Index);

解释：

按位读取通用输入点 I1-I20 状态。

轴正在运行时是否可调用：是

参数：

Board_NO : 0－3, 板号
Index : 1 - 20, 输入点索引号

返回码：

0-1 输入点状态
(-1) 不成功

参考函数：

ReadIO_9010
WriteIo_9010

§3.9.4 WriteIo_9010

置板上数字 IO（输出）点信号

函数编号:16

函数原型：

C: short APIENTRY WriteIo_9010(unsigned short Board_NO,unsigned short IO_V);

解释：

这个函数将置板上 8 路通用数字输出点状态。达林顿输出方式： 1：导通；0：断开。

轴正在运行时是否可调用：是

参数：

Board_NO 卡号，可能值 0，1，2，3
IO_V 输出点值,WORD 型，低 8 位有效,对应 O1-O8,8 个输出点状态

返回码：

(0) 成功
(-1) 参数非法

参考函数：

ReadIO_9010

§3.9.5 WriteIoBit_9010

按位置板上数字 IO（输出）点信号

函数编号:81

函数原型：

C: short APIENTRY WriteIoBit_9010(unsigned short Board_NO,unsigned short IO_V,short Index);

解释：

这个函数将置板上 8 路通用数字输出点状态。达林顿输出方式： 1：导通；0：断开。

轴正在运行时是否可调用：是

参数：

Board_NO ： 卡号，可能值 0，1，2，3
IO_V ： 0 - 1，输出点值
Index ： 1 - 8, 输出点索引号

返回码：

(0) 成功
(-1) 参数非法

参考函数：

ReadIO_9010

§3.9.6 ReadOs_9010

读取输出点 O1-O8 状态

函数编号:82

函数原型：

C: short APIENTRY ReadOs_9010(unsigned short Board_NO);

解释：

读取通用输出点 O1-O8 状态, 返回数值转换为二进制时的 0-7 位有效。由于可以用插补模式来设置输出点（函数 LM_IOOut_9010），而该模式的输出点状态输出不是立即生效的，所以可以用该函数来随时检测输出点状态。

轴正在运行时是否可调用：是

参数：

Board_NO ： 卡号，可能值 0，1，2，3

返回码：

0-256	低 8 位有效,对应 O1-O8,8 个输出点状态
(-1)	参数非法

参考函数：

ReadOsBit_9010

§3.9.7 ReadOsBit_9010

按位读取输出点 O1-O8 状态

函数编号:83

函数原型：

C: short APIENTRY ReadOsBit_9010(unsigned short Board_NO,short Index);

解释：

按位读取通用输出点 O1-O8 状态。由于可以用插补模式来设置输出点，由于可以用插补模式来设置输出点（函数 LM_IOOut_9010），而该模式的输出点状态输出不是立即生效的，所以

可以用该函数来随时检测输出点状态。

轴正在运行时是否可调用：是

参数：

Board_NO ： 卡号，可能值 0，1，2，3
Index ： 1 - 8, 输出点索引号

返回码：

0 - 1 对应输出点状态
(-1) 参数非法

参考函数：
ReadOs_9010

§3.9.8 ReadMPGIO_9010

读手轮 IO 输入点 MPG_I1-MPG_I7 状态
函数编号:118
函数原型：

C: long APIENTRY ReadMPGIO_9010(unsigned short Board_NO,short Index);

解释：
读取附加编码器输入点 I1-I7 状态。硬件部分参看 ***§1.7 附加编码器接口*** 节

轴正在运行时是否可调用：是

参数：
Board_NO : 0－3, 板号
Index : 0=全读，返回值的 bit0-bit6 对应 MPG_I1-MPG_I7 状态，1-7=按位读取，返回对应位状态

返回码：
Index=0 时 0-6 位有效, 对应输入点 MPG_I1-MPG_I7 状态，Index=1-7 时，返回对应位状态（0 或 1）
-1=不成功,

参考函数：

ReadIOBit_9010

WriteIo_9010

§3.10 附加编码器控制函数

§3.10.1 ReadEncoderPos_9010

读取编码器位置

函数编号:10

函数原型：

C: long APIENTRY ReadEncoderPos_9010(unsigned short Board_NO);

解释：

该函数返回附加编码器计数器值

轴正在运行时是否可调用：是

参数：

卡号，可能值 0，1，2，3

返回码：

编码器位置； 范围；长整型数； 单位， 脉冲个数。

参考函数：

HomeEncoder_9010

§3.10.2 HomeEncoder_9010

附加编码器位置清零

函数编号:11

函数原型：

C:　　short APIENTRY HomeEncode_9010(unsigned short Board_NO);

解释：

该函数将附加编码器位置计数器清零。可以通过调用函数 ReadEncoderPos_9010 来得到编码器当前位置计数器。

轴正在运行时是否可调用：当有轴跟附加随编码器时不能，其它情况下可以

参数：

卡号，可能值 0，1，2，3

返回码：

(0)　　成功

(-1)　　不成功

参考函数：

ReadEncoderPos_9010

§3.10.3　SetEncodeCount_9010

设附加编码器初值

函数编号:113

函数原型：

C:　　short APIENTRY SetEncodeCount_9010(unsigned short Board_NO,long Offset);

解释：

该函数设附加编码器位置计数器初值。可以通过调用函数 ReadEncoderPos_9010 来得到编码器当前位置计数器。

轴正在运行时是否可调用：当有轴跟附加随编码器时不能，其它情况下可以

参数：

卡号，可能值 0，1，2，3

offset　：编码器初值; 范围: 长整型数;　单位: 脉冲个数。

返回码：

(0)　　成功

(-1)　　不成功

参考函数：

ReadEncoderPos_9010

§ 3.11　PWM 脉冲/DA 输出控制

9010 有一路附加的 PWM/DA 输出源，硬件设置部分参看 ***§ 1.6 Vout/PWM 输出选择***

§ 3.11.1　PwmOut_9010

PWM 脉冲输出

函数编号:12

函数原型：

C:　　short APIENTRY PwmOut_9010(unsigned short Board_NO,long frequency,float Pulse_Highf);

解释：

9010 卡上有一路 PWM 输出，调用该函数即可使其输出设定的频率和占空比的 PWM 信号。

轴正在运行时是否可调用：是

参数：

Board_NO :　0－3, 板号

frequency:　18-1500000;　PWM 输出频率; 单位: 赫兹(Hz)

Pulse_Highf:　占空比,范围:0.0-1.0;　精度:不小于 1%

返回码：

(0)　成功

(-1)　参数非法

参考函数：

PwmOut2_9010

PwmStop_9010

DAOut_9010

LM_PWM_9010

§3.11.2 PwmOut2_9010

PWM 脉冲输出

函数编号:12

函数原型：

C: short APIENTRY PwmOut2_9010(unsigned short Board_NO,long frequency,float Pulse_Highf);

解释：

9010 卡上有一路 PWM 输出，调用该函数即可使其输出设定的频率和占空比的 PWM 信号。

轴正在运行时是否可调用：是

参数：

Board_NO : 0－3, 板号

frequency: 18-1500000; PWM 输出频率; 单位: 赫兹(Hz)

Pulse_Highf: 高电平脉冲宽度,单位：毫秒;

精度（分辨率）:0.00000667 当 frequency≥2288 ；

0.000853 当 frequency <2288

返回码：

(0) 成功

(-1) 参数非法

参考函数：

PwmOut_9010

PwmStop_9010

DAOut_9010

LM_PWM_9010

§3.11.3 PwmStop_9010

PWM 输出停止

函数编号:13

函数原型：

C: short APIENTRY PwmStop_9010(unsigned short Board_NO);

解释：

停止 PWM 输出。

轴正在运行时是否可调用：是

参数：

Board_NO 卡号，可能值 0，1，2，3

返回码：

(0) 成功

(-1) 参数非法

参考函数：

PwmOut_9010

PwmOut2_9010

DAOut_9010

LM_PWM_9010

§3.11.4 DAOut_9010

DA(数模转换)模拟量输出

函数编号:68

函数原型：

C: short APIENTRY DAOut_9010(unsigned short Board_NO,double DA_Avlue);

解释：

轴正在运行时是否可调用：是

参数：

Board_NO : 0－3, 板号

DA_Avlue : -10 - +10 V; 精度: 1/6000; 即: 精度大于 12 位(4096),不到 13 位(8192).
该 DA 输出与 PWM 输出占用同一个硬件资源,通过 9010 端子板上的跳线选择是 DA 输出还是 PWM 输出.

返回码：

(0) 成功

(-1) 参数非法

参考函数：

PwmOut_9010

PwmStop_9010

LM_PWM_9010

§3.12 插补函数

§3.12.1 插补初始化函数

§3.12.1.1 LM_SetXAxis_9010

将实际轴与插补引擎的 X 轴相匹配

函数编号:32

函数原型：

C： short APIENTRY LM_SetXAxis_9010(unsigned short Board_NO,short Axis_NO,double factor_c_t,double delta);

解释：

该函数将实际轴与插补引擎虚拟的 X 轴相匹配。插补引擎虚拟了一个 XYZW 坐标系，在执

行插补运算前，用户必须将这个虚拟的 XYZW 坐标系实际化，可对虚拟的 XYZW 坐标的任意两轴直至四轴全部实际化。

同理：也有 YZW 轴的匹配函数：

short LM_SetYAxis_9010 (short Board_NO,short Axis_NO,
double factor_c_t,double delta);
short LM_SetZAxis_9010 (short Board_NO,short Axis_NO,
double factor_c_t,double delta);
short LM_SetWAxis_9010 (short Board_NO,short Axis_NO,
double factor_c_t,double delta);

在插补引擎正在运行当中是否可调用：否

参数：

Board_NO: 卡号，可能值 0，1，2，3

axis_NO： 轴号，可能值为：0，1，2，3。

factor_c_t: 轴的脉冲当量。即：电机转动一个脉冲的步距角时，该轴移动多少用户单位。

例如 1，某一直线轴通过滚珠丝杠与步进电机直联，该滚珠丝杠的螺距是 5 毫米，而电机转一圈需 2000 个脉冲，则：轴的脉冲当量=5/2000=0.0025；
0.0025 的含义是：一个脉冲将使该轴移动 0.0025 毫米。

例如 2，某一直线轴通过滚珠丝杠与伺服电机直联，该滚珠丝杠的螺距是 5 毫米，而伺服电机编码器为 2500 线，由于四倍频或伺服电机电子齿轮设定的关系，伺服电机转一圈需 10000 个脉冲，则：轴的脉冲当量=5/10000=0.0005；
0.0005 的含义是：一个脉冲将使该轴移动 0.0005 毫米。

轴的脉冲当量很重要，轴的脉冲当量确定后，本书以下有关轴的直线、圆弧插补的距离单位就确定。上例 1 中如果：factor_c_t=0.0025，则该轴的距离单位就是毫米。

delta： 该轴动态准确定位的误差值。单位为用户单位。

一般来说，该值不能小于轴的脉冲当量，以轴的脉冲当量的 3-5 倍为宜。

在插补运算时，在每行插补运动开始之前，系统将以该值乘于 10 倍为依据来判断轴位置误差是否超限，如果超越极限，插补引擎将停止工作并报警。报警号为 3（插补误差超限）。

返回码：

(0)　　成功
(-1)　　不成功

参考函数：

LM_OffXAxis_9010

§3.12.1.2　LM_OffXAxis_9010

撤消插补引擎的 X 轴匹配

函数编号:33

函数原型：

C：　　short APIENTRY LM_OffXAxis_9010(unsigned short Board_NO);

解释：

该函数将撤消插补引擎的 X 轴匹配。它是函数 LM_SetIpol_X_Axis9010 的反操作。
同理：也有 YZW 轴的撤消匹配函数：

short LM_OffYAxis_9010(short Board_NO);
short LM_OffZAxis_9010(short Board_NO);
short LM_OffWAxis_9010(short Board_NO);

在插补引擎正在运行当中是否可调用：否

参数：

Board_NO　　卡号，可能值 0，1，2，3

返回码：

(0)　　成功
(-1)　　不成功

参考函数：

LM_SetIpol_X_Axis9010

§3.12.1.3 LM_GetXStartPos_9010

获得插补轴 X 当前位置

函数编号:54

函数原型：

C： short APIENTRY LM_GetXStartPos_9010(unsigned short Board_NO,double *Pos);

解释：

指示 9010 卡，轴的现在位置是轴插补的开始位置，并可获得轴当前位置（用户单位），在发送插补命令(LM_Line_9010, LM_ArcCCW_9010)开始前调用，类似的函数还有：

short APIENTRY LM_GetYStartPos_9010(unsigned short Board_NO,double *Pos);

short APIENTRY LM_GetZStartPos_9010(unsigned short Board_NO,double *Pos);

short APIENTRY LM_GetWStartPos_9010(unsigned short Board_NO,double *Pos);

注意：在该命令之前，先写入 LM_CleanBuff_9010 （清空插补命令缓冲区）命令

在插补引擎正在运行当中是否可调用：否

参数：

Board_NO : 0－3, 板号

Pos : 获得 轴插补的开始位置,单位为用户单位。指针类参数,可以为 NULL(空指针) ，VB 用户可以直接设为 0，或用同等函数 LM_GetAxisStartPos_9010。

返回码：

(0) 成功

(-1) 不成功

参考函数：

LM_GetAxisStartPos_9010

§3.12.1.4 LM_GetAxisStartPos_9010

获得插补轴当前位置

函数编号:54

函数原型：

C： double APIENTRY LM_GetAxisStartPos_9010(unsigned short Board_NO,short AxisFlag);

解释：

指示 9010 卡，轴的现在位置是轴插补的开始位置，并获得轴当前位置（用户单位），在发送插补命令(LM_Line_9010, LM_ArcCCW_9010)开始前调用。

注意：如果在用户程序中多次用到上述命令，在第二次用该命令之前，先写入 LM_CleanBuff_9010 **（清空插补命令缓冲区）命令**

在插补引擎正在运行当中是否可调用：否

参数：

Board_NO : 0－3, 板号

AxisFlag : 1- 4, 指示是那个轴；1：X 轴， 2：Y 轴， 3：Z 轴， 4：W 轴。

返回码：

轴的当前位置。用户单位

当返回轴位置=-2147483648 时,表示不成功,有错误产生。

参考函数：

LM_GetXStartPos_9010

§3.12.2 插补顺序执行函数

§3.12.2.1 LM_Line_9010

写入直线插补命令

函数编号:42

函数原型：

C： short APIENTRY LM_Line_9010(unsigned short Board_NO,double xPos,double yPos,double zPos,double wPos,double Speed,long LineNO);

解释：

该函数将直线插补命令写入插补缓冲区。

在插补引擎正在运行当中是否可调用：是

参数：

Board_NO : 0－3, 板号

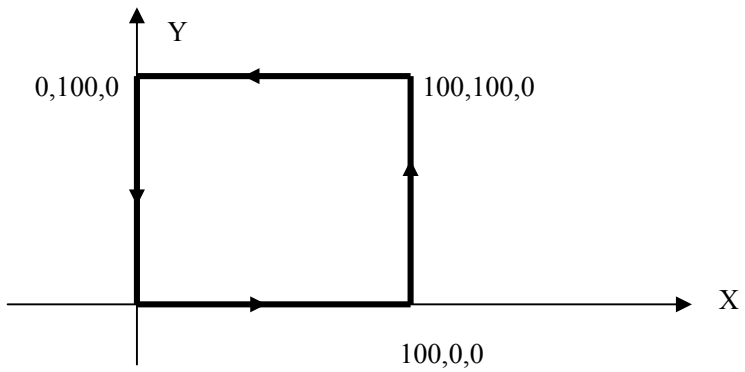
xPos、yPos、zPos、wPos： XYZW 轴的终点位置；绝对坐标，单位为用户单位。

Speed： 插补速度，单位：用户单位/分钟。

LineNO： 用户自定义行号。当插补引擎运行到该行时，用 LM_GetWorkLine9010 (long *LineNO)函数可检测该行号值。

例 1 直线插补（C 语言）：

设 XYZ 轴从当前位置（0，0，0，0）直线移动到（100，0，0，0）位置，再直线移动到（100，100，0，0）位置，再直线移动到（0，100，0，0）位置，再直线移动到（0，0，0，0）位置；如下图：



插补速度 F=200，行号从 100 开始；

则：

```
double F=200;
long   LineNO=100;
LM_Line_9010 (0, 100，0，0，0，F，LineNO)；
LM_Line_9010 (100，100，0，0，F，LineNO+1)；
LM_Line_9010 (0，100，0，0，F，LineNO+2)；
LM_Line_9010 (0，0，0，0，F，LineNO+3)；
LM_End_9010(0,LineNO+4);
LM_Start_9010(0, LineNO+5);
```

返回码：

(0)　　成功
(-1)　　不成功

参考函数：

LM_ArcCW_9010

LM_End_9010

LM_Start_9010

§3.12.2.2　LM_LineMaxV_9010

直线插补

函数编号:69

函数原型：

C：　　short APIENTRY LM_LineMaxV_9010(unsigned short Board_NO,double xPos,double yPos,double zPos,double wPos,double Speed,double MaxSpeed,long LineNO);

解释：

该函数与 LM_Line_9010 区别是：①可单独指定本行的最大速度，②执行到该行时不会有 PWM 信号输出（如果用户用到 LM_PWM_9010 函数）。

在插补引擎正在运行当中是否可调用：是

参数：

Board_NO : 0－3, 板号
xPos　　: X 轴插补的终点位置(绝对值),单位为用户单位。
yPos　　: y 轴插补的终点位置(绝对值),单位为用户单位。
zPos　　: z 轴插补的终点位置(绝对值),单位为用户单位。
wPos　　: w 轴插补的终点位置(绝对值),单位为用户单位。
Speed　　: 插补速度,单位: 用户单位/分钟
MaxSpeed : 本行插补最大速度,单位: 用户单位/分钟
LineNO　　: 插补行号,用户自由设定

返回码：

(0)　　成功
(-1)　　不成功

参考函数：

§ 3.12.2.3　LM_LineMeasure_9010

直线插补及测量 IO 点输入

函数编号:65

函数原型：

C：　short APIENTRY LM_LineMeasure_9010(unsigned short Board_NO,double xPos,double yPos,double zPos,double wPos,short IO_Index,short Mode,double Speed,long LineNO);

解释：

直线插补并检测由参数 IO_Index 指示的输入点状态，如检测到输入点，就立即停止。该函数与 LM_Line_9010 区别是：

①本行的最大速度就是由 Speed 指定（函数 LM_SetSpeedRate_9010 不能将其增速）。
②执行到该行时不会有 PWM 信号输出（如果用户用到 LM_PWM_9010 函数）。
③当 9010 卡检测到由 IO_Index 指定的输入点后立即停止（轴停止模式为 CeaseAxis_9010）。
④在本行执行完之前，不能再加入新的插补函数，如果加入，9010 卡将报错，错误号为 18。
⑤在本行执行当中，如果检测到输入点（由参数 IO_Index 指示），插补运动立即停止，这时插补状态（由函数 LM_GetState_9010 获得）将为 3（被 LM_End_9010 结束），函数 LM_GetMeasureState_9010 将返回状态 1。
⑥在本行执行当中，如果没有检测到输入点，在本行执行完后插补运动也将停止，这时插补状态（由函数 LM_GetState_9010 获得）将为 4（被 LM_End_9010 结束），函数 LM_GetMeasureState_9010 将返回状态 0。
⑦无论检测到输入点与否，在本行执行完后，如果还要进行插补运动，都要进行如下操作：

```
LM_CleanBuff_9010(…);          //清除插补缓存
LM_GetXStartPos_9010(…);       //重新获得插补开始位置
LM_GetYStartPos_9010(…);
LM_GetZStartPos_9010(…);
LM_GetWStartPos_9010(…);
```

在插补引擎正在运行当中是否可调用：是

参数：

Board_NO : 0－3，板号
xPos　　: X 轴插补的终点位置(绝对值),单位为用户单位。

yPos : y 轴插补的终点位置(绝对值),单位为用户单位。

zPos : z 轴插补的终点位置(绝对值),单位为用户单位。

wPos : w 轴插补的终点位置(绝对值),单位为用户单位。

IO_Index : 1-8; 通用 IO 点输入点序号,

Mode : 1-4;

1：通用 IO 点输入点为 1 时，立即停止(CeaseAxis_9010 模式)。

2：通用 IO 点输入点为 1 时，立即停止(AbortAxis_9010 模式)。

3：通用 IO 点输入点为 0 时，立即停止(CeaseAxis_9010 模式)。

4：通用 IO 点输入点为 0 时，立即停止(AbortAxis_9010 模式)。

Speed : 插补速度,单位: 用户单位/分钟

LineNO : 插补行号,用户自由设定

返回码：

(0) 成功

(-1) 不成功

参考函数：

LM_GetMeasureState_9010

§3.12.2.4 LM_GetMeasureState_9010

获得当前测量状态

函数编号:66

函数原型：

C： short APIENTRY LM_GetMeasureState_9010(unsigned short Board_NO);

解释：

该函数将获得当前测量状态（由插补命令 LM_LineMeasure_9010 产生）。

在插补引擎正在运行当中是否可调用：是

参数：

Board_NO : 0－3, 板号

返回码：

0:　　没有检测到测量信号

1:　　检测到测量信号

255:　　命令不成功

注意：检测到输入点后，如果有新的直线插补测量运动开始执行，测量状态将自动转为 0 状态。

参考函数：

§ 3.12.2.5　LM_LineFE_9010

插补模式的 轴跟随编码器位置命令

函数编号:77

函数原型：

C：　short APIENTRY LM_LineFE_9010(unsigned short Board_NO,double xPos,double yPos,double zPos,double wPos,double Rate,double Kp,short Mode,double Speed,long LineNO);

解释：

插补模式下的轴跟随编码器位置命令。该函数将 写入插补缓冲区。该函数的主要目的是为了实现螺纹切削。以数控铣床为例，编码器同轴安装在主轴上，用该函数可实现硬攻丝。

在插补引擎正在运行当中是否可调用：是

参数：

Board_NO :　　0－3, 板号

xPos、yPos、zPos、wPos：　　XYZW 轴的终点位置；绝对坐标，单位为用户单位。

XYZW 轴中只有一个轴可以实现插补模式的编码器位置跟随，9010 卡根据各轴起点与终点位置之差是否为 0，自动判断那个轴跟随。如果有多于一个轴的位置之差不为 0，9010 卡将报错。

Mode　　: 0,1; 0=到达目标,立即停止跟随; 0=到达目标,不停止跟随,由后续命令决定;

当 Mode=0 时，跟随轴到达或超过终点位置时，跟随轴将立即停止（以 AbortAxis_9010 模式）。当 Mode=1 时，跟随轴到达或超过终点位置时，将由后续命令决定，细节如下：

直线插补命令：同轴同方向，跟随轴将不停，自动过渡到直线插补；其它情况下，跟随轴将立即停止（以 AbortAxis_9010 模式）。

圆弧插补命令：跟随轴将立即停止（以 AbortAxis_9010 模式）。

其它插补命令：（包括 LM_End_9010 命令）不停止跟随,再由后续命令决定。

Rate ：轴跟随比率：

设：

A1=跟随轴开始位置，A2=跟随轴实时位置，单位：用户单位

B1=编码器开始位置，B2=编码器实时位置，单位：脉冲数

则：

$$Rate = \frac{A2 - A1}{B2 - B1}$$

可正负; 范围: 绝对值

$0.001 \times$跟随轴脉冲当量 至 $1000 \times$跟随轴脉冲当量;

单位: 无; 分辨率: $0.001 \times$跟随轴脉冲当量

B1=编码器开始位置：

9010 卡根据前面是否有未完成的***轴跟随编码器位置命令*** 对编码器进行清零操作。如果没有，9010 卡将自动清零编码器计数器，防止计数器溢出。是否自动清零可由函数 GetEnAuto0Flag_9010 探知。

Kp ：跟随 PID 调解系数; 范围: 0.001-1000; 单位: 无; 分辨率: 0.001

Speed ：插补预期速度,也应该是本行插补最大速度,单位: 用户单位/分钟

实际该行的插补速度将由 Rate 参数和编码器所反馈的速度决定。

LineNO ：插补行号,用户自由设定

例 1 用该函数实现数控铣床的硬攻丝（C 语言）：

设主轴编码器线数为 2500 线，在 9010 卡中将四倍频，为：10000 线；螺距为 0.8；插补速度 F=200，行号从 100 开始；

则：

```
double F=200;
long   LineNO=100;
double Rate=-1*0.8/10000;
```

LM_Line_9010 (0，0，0，0，0，F，LineNO)；

LM_Line_9010 (0，0，-5，0，F，LineNO+1)；

LM_LineFE_9010 (0，0，-25，0，1，Rete，0.1，F，LineNO+2)； //Z 轴-5 至-25 攻丝

LM_IOOut_9010 (0,……); //控制主轴反转

LM_LineFE_9010 (0，0，-5，0，0，Rete，0.1，F，LineNO+4)； // Z 轴-25 至-5 回退

LM_End_9010(0,LineNO+5);

LM_Start_9010(0, LineNO+6);

返回码：

(0) 成功

(-1) 不成功

参考函数：

LM_ArcCW_9010

LM_End_9010

LM_Start_9010

§ 3.12.2.6 LM_ArcCW_9010

写入圆弧/螺旋线插补命令

函数编号:43、44

函数原型：

C short APIENTRY LM_ArcCW_9010(unsigned short Board_NO,double xPos,double yPos,double zPos,double wPos,double xcPos,double ycPos,double Speed,long LineNO);

解释：

该函数将圆心在 xy 平面的顺时针圆弧/螺旋线插补命令写入插补缓冲区。同理，还有另外两个类似的顺时针插补命令：

short LM_ArcCW_YZ_9010 (short Board_NO,

double xPos,double yPos,double zPos,double wPos,

double ycPos,double zcPos,

double Speed,long LineNO);

short LM_ArcCW_ZX_9010 (short Board_NO,
double xPos,double yPos,double zPos,double wPos,
double zcPos,double xcPos,
double Speed,long LineNO);

以及三个逆时针插补命令：

short LM_ArcCCW_9010(short Board_NO,
double xPos,double yPos,double zPos,double wPos,
double xcPos,double ycPos,
double Speed,long LineNO);

short LM_ArcCCW_YZ_9010 (short Board_NO,
double xPos,double yPos,double zPos,double wPos,
double ycPos,double zcPos,
double Speed,long LineNO);

short LM_ArcCCW_ZX_9010 (short Board_NO,
double xPos,double yPos,double zPos,double wPos,
double zcPos,double xcPos,
double Speed,long LineNO);

在插补引擎正在运行当中是否可调用：是

参数：

Board_NO : 0－3, 板号

XY 平面：

xPos、yPos、zPos、wPos： XYZW 轴的终点位置；绝对坐标，单位为用户单位。

xcPos : X 轴的圆心位置(绝对值),单位为用户单位。

ycPos : Y 轴的圆心位置(绝对值),单位为用户单位。

YZ 平面：

xPos、yPos、zPos、wPos： XYZW 轴的终点位置；绝对坐标，单位为用户单位。

ycPos : Y 轴的圆心位置(绝对值),单位为用户单位。

zcPos : Z 轴的圆心位置(绝对值),单位为用户单位。

ZX 平面：

xPos、yPos、zPos、wPos：　　XYZW 轴的终点位置；绝对坐标，单位为用户单位。

zcPos　　：　Z 轴的圆心位置(绝对值),单位为用户单位。

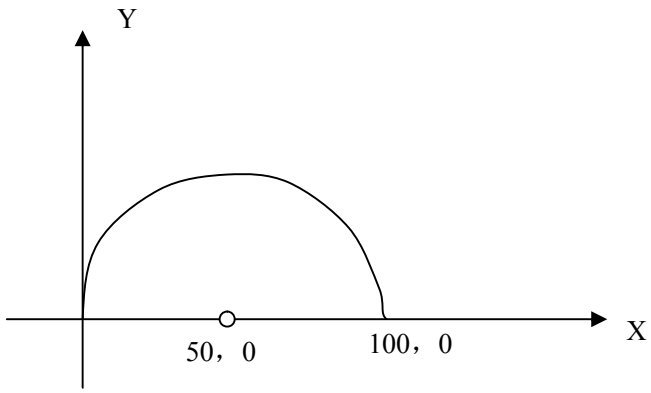
xcPos　　：　X 轴的圆心位置(绝对值),单位为用户单位。

Speed：　　插补速度，单位：用户单位/分钟。

LineNO：　　用户自定义行号。

例 1（C 语言）：

设 XYZ 轴的当前位置为 0、0、0、0；要运行一个在 XY 平面的顺圆如下图：

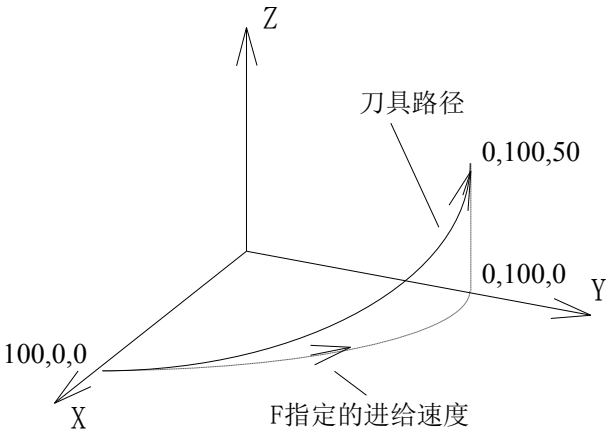


插补速度是：500，行号为 100；

则： LM_ArcCW_9010 (0，100，0，0，0，50，0，500，100);

例 2 螺旋线插补（C 语言）：

设 XYZW 轴从当前位置（0，0，0，0）直线移动到（100，0，0，0）位置，再以螺旋线轨迹移动到终点位置（0，100，50）；如下图：



插补速度 F=200，行号从 100 开始；

则：

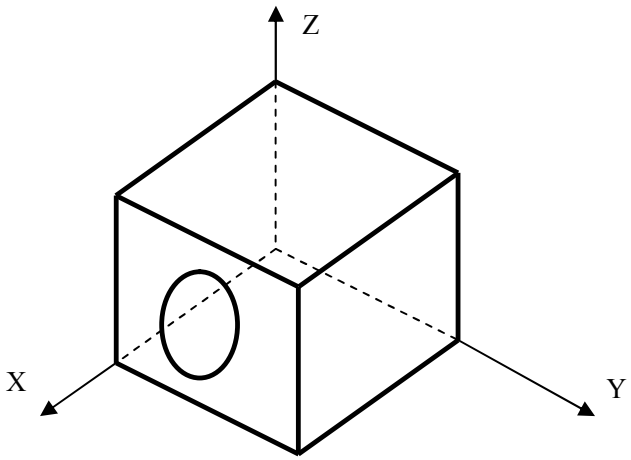
```
double F=200;
long   LineNO=100;
LM_Line_9010 (0，100，0，0，0，F，LineNO)；
LM_ArcCW_9010 (0，0，100，50，0，0，0，F，LineNO+1)；
LM_End_9010(0, LineNO+2);
    LM_Start_9010(0,0);
```

由F值指定的插补速度是指沿着圆弧的进给速度，直线轴的进给速度为：

$F_{(直线)}$=F*直线轴的长度/圆弧的弧长。

决定插补速度F值时要十分注意直线轴的进给率不要超过各种限制。

在上例中，圆弧插补均在 XY 平面完成，实际应用中有可能在 YZ 平面进行插补运动，比如在数控加工中心的机床上加工箱体零件，在箱体侧面铣一个圆孔，这时就会用到在不同平面的圆弧插补：



需要注意的是，XY 平面、YZ 平面、ZX 平面的圆弧插补函数，轴终点位置参数、圆心坐标参数，其代入次序各不相同，见参数说明。

返回码：

大于 0　成功

小于 0　非法参数

参考函数：

LM_Line_9010

LM_End_9010

LM_Start_9010

§3.12.2.7 LM_Wait_9010

插补等待

函数编号:60

函数原型：

C： short APIENTRY LM_Wait_9010(unsigned short Board_NO,unsigned long Millisecond,long LineNO);

解释：插补等待

该命令将把插补数据送入 9010 卡的插补缓存器中, 9010 卡的插补缓存器总共有 265 行. 当插补引擎运行到该行时将暂停运行, 在等待用户所设时间后, 再开始下一步插补命令。

在插补引擎正在运行当中是否可调用：是

参数：

Board_NO : 0－3, 板号

Millisecond : 等待时间,毫秒

LineNO : 插补行号,用户自由设定

返回码：

(0) 成功

(-1) 不成功

参考函数：

§3.12.2.8 LM_PWM_9010

插补模式的 PWM 输出

函数编号:61

函数原型：

C: short APIENTRY LM_PWM_9010(unsigned short Board_NO,long frequency,double Pulse_Highf,double Speed,long LineNO);

解释：

插补模式的 PWM 输出，占空比随速度变化。该命令将把插补数据送入 9010 卡的插补缓存器中,9010 卡的插补缓存器总共有 256 行。当插补引擎运行到该行时将根据参数决定 PWM 输出.

轴正在运行时是否可调用：是

参数：

Board_NO　　: 0－3, 板号

frequency　　:18-1500000;　　PWM 输出频率; 单位: 赫兹(Hz)

当为 0 时,停止 PWM 输出

Pulse_Highf　: 占空比,范围:0.0-1.0;　精度:不小于 1%

该值对应插补速度达到 Speed 所设速度时的最大占空比,当插补速度大于 Speed 所设速度时,也按该参数输出.

Speed　　: 占空比随动的最大插补速度,单位: 用户单位/分钟

当该值为 0 时,表明不是随动模式,当插补引擎运行到该行时将根据 frequency,Pulse_Highf 参数值立即 PWM 输出.

LineNO　　: 插补行号,用户自由设定

返回码：

(0)　成功

(-1)　不成功

§3.12.2.9　LM_IOOut_9010

插补模式的 IO 点输出

函数编号:62

函数原型：

C:　short APIENTRY LM_IOOut_9010(unsigned short Board_NO,short IO_Index,short IO_Value,long LineNO);

解释：

该命令将把插补数据送入 9010 卡的插补缓存器中, 9010 卡的插补缓存器总共有 256 行. 当插补引擎运行到该行时将根据参数输出 I0 点.

在插补引擎正在运行当中是否可调用：是

参数：

Board_NO　　: 0－3, 板号

IO_Index　　: 1-8;　通用 IO 点输出点序号,

IO_Value　　: 0 - 1; 输出点值

LineNO　　：插补行号,用户自由设定

返回码：

(0)　　成功

(-1)　　不成功

参考函数：

§3.12.2.10　LM_Wait_I_9010

插补等待通用 I0 点输入

函数编号:63

函数原型：

C：　　short APIENTRY LM_Wait_I_9010(unsigned short Board_NO,short IO_Index,long LineNO);

解释：

该命令将把插补数据送入 9010 卡的插补缓存器中, 9010 卡的插补缓存器总共有 256 行. 当插补引擎运行到该行时将等待，直到输入点为 1 时再执行下一行插补命令.

在插补引擎正在运行当中是否可调用：是

参数：

Board_NO　　: 0－3, 板号

IO_Index　　: 1-8;　通用 IO 点输入点序号,

LineNO　　：插补行号,用户自由设定

返回码：

(0)　　成功

(-1)　　不成功

参考函数：

§3.12.2.11 LM_End_9010

写入插补结束命令

函数编号:45

函数原型：

C： short APIENTRY LM_End_9010(unsigned short Board_NO,long LineNO);

解释：

该命令将把插补数据送入 9010 卡的插补缓存器中,9010 卡的插补缓存器总共有 256 行。当插补引擎运行到该行时将停止运行,如要重新开始,则再送入插补数据命令(LM_Line_9010,LM_ArcCW_9010,LM_ArcCCW_9010),并发出 LM_Start_9010 命令

在插补引擎正在运行当中是否可调用：是

参数：

Board_NO : 0－3, 板号

LineNO : 用户自定义行号。当插补引擎运行到该行时，用 LM_GetLineNO_9010(short Board_NO);函数可检测该行号值。

返回码：

(0) 成功

(-1) 不成功

参考函数：

LM_Line_9010

LM_ArcCW_9010

LM_Start_9010

§3.12.3 插补控制函数

§3.12.3.1 LM_SetACCDec_9010

设置插补加速度和插补减速度

函数编号:40

函数原型：

C： short APIENTRY LM_SetACCDec_9010(unsigned short Board_NO,long acceleration,long deceleration);

解释：

该函数将设置插补加速度和插补减速度。

在插补引擎正在运行当中是否可调用：否

参数：

Board_NO	0－3, 板号
acceleration	插补加速度，用户单位/秒平方。范围：1-10000。缺省值: 500
Deceleration	插补减速度，用户单位/秒平方。范围：1-10000。缺省值: 500

返回码：

0	成功
(<0)	不成功

参考函数：

LM_SetSAccePower_9010

§ 3.12.3.2 LM_SetSAccPower_9010

设置插补 S 型加速度(1,2,3,4)指数

函数编号:71

函数原型：

C： short APIENTRY LM_SetSAccePower_9010(unsigned short Board_NO,short PowerFlag);

解释：

（保留函数）

在插补引擎正在运行当中是否可调用：否

参数：

Board_NO　：　0－3, 板号

PowerFlag　：　1-4　S 型加速度(1,2,3,4)指数，1=直线形加速度。系统缺省值：2

返回码：

0　　成功

(<0)　　不成功

参考函数：

LM_SetACCDec_9010

§3.12.3.3　LM_Start_9010

启动插补引擎，插补开始

函数编号:41

函数原型：

C：　　short APIENTRY LM_Start_9010(unsigned short Board_NO,short ObligeFlag);

解释：

在用户写入一定数量的插补命令后，可调用该函数让插补引擎开始工作。

在插补引擎正在运行当中是否可调用：是

参数：

Board_NO ：　　卡号，可能值 0，1，2，3

ObligeFlag：　　强制运行插补命令标志。0=不强制；1=强制。根据第 ***§2.5.7 插补疑问*** 节所介绍内容，当用户写入 1 或 2 行短线段插补命令时，可在随后的 LM_Start_9010　命令中将该参数置为 1，强制运行插补命令。

返回码：

(0)　　成功

(-1)　　不成功

参考函数：

LM_Line_9010
LM_ArcCW_9010
LM_End_9010

§3.12.3.4 LM_CleanBuff_9010

清空插补命令缓冲区
函数编号:46
函数原型：

C： short APIENTRY LM_CleanBuff_9010(unsigned short Board_NO);

解释：

这个函数将清空插补命令缓冲区。

在插补引擎正在运行当中是否可调用：否

参数：
Board_NO : 0－3, 板号

返回码：

(0) 成功
(-1) 不成功

参考函数：

§3.12.3.5 LM_GetBuffLen_9010

获得插补命令缓冲区空闲长度
函数编号:55
函数原型：

C： short APIENTRY LM_GetBuffLen_9010(unsigned short Board_NO);

解释：

这个函数将返回插补缓冲区的空闲长度。控制卡有 256 行命令缓冲区。调用该函数还有附加功能，详细说明请看第 ***§2.5.7 插补疑问*** 节所介绍内容。

在插补引擎正在运行当中是否可调用：是

参数：

返回码：

0-256　成功，插补缓存长度
(-1)　不成功

参考函数：

§3.12.3.6 LM_SetMinVel_9010

设置插补最小速度

函数编号:47

函数原型：

C：
```
short APIENTRY LM_SetMinVel_9010(unsigned short Board_NO,
                                 double MinLineVel,
                                 double MinArcVel);
```

解释：

在插补引擎正在运行当中是否可调用：否

参数：

Board_NO　: 0－3, 板号
MinLineVel: 直线最小插补速度, 单位: 用户单位/分钟, 缺省值: 30
MinArcVel : 圆弧最小插补速度, 单位: 用户单位/分钟, 缺省值: 30

返回码：

(0)　　成功

(-1)　　不成功

参考函数：

LM_SetMaxVel_9010

§3.12.3.7　LM_SetMaxVel_9010

设置插补最大速度

函数编号:57

函数原型：

C：　short APIENTRY LM_SetMaxVel_9010(unsigned short Board_NO,
double MaxLineVel,
double MaxArcVel);

解释：

在插补引擎正在运行当中是否可调用：否

参数：

Board_NO　: 0－3, 板号

MaxLineVel: 直线最大插补速度, 单位: 用户单位/分钟, 缺省值: 40000

MaxArcVel : 圆弧最大插补速度, 单位: 用户单位/分钟, 缺省值: 40000

返回码：

(0)　　成功

(-1)　　不成功

参考函数：

LM_SetMinVel_9010

§3.12.3.8 LM_SetParaAngle_9010

设置插补引擎的角度参数

函数编号:48

函数原型：

C： short APIENTRY LM_SetParaAngle_9010(unsigned short Board_NO,short angle1,short angle2);

解释：

该函数将设置插补引擎内部参数。这些参数将影响两两直线插补线段之间的运动平滑性。具体含义请看第 ***§2.5.7 插补疑问*** **（4）两线段之间如何进行插补速度衔接** 节所介绍的内容。

在插补引擎正在运行当中是否可调用：否

参数：

Board_NO : 0－3, 板号

ang1：: 值范围：0-90，且 ang1<ang2。插补引擎初始化时该参数的却省值为 20。

ang2：: 值范围：0-90，且 ang1<ang2。插补引擎初始化时该参数的却省值为 35。

返回码：

大于 0 成功

小于 0 非法参数

参考函数：

§3.12.3.9 LM_GetLineNO_9010

获得当前插补行号

函数编号:49

函数原型：

C： long APIENTRY LM_GetLineNO_9010(unsigned short Board_NO);

解释：

这个函数将获得插补引擎当前运行的插补行号，该行号是用户在写入插补命令时自定义的。

在插补引擎正在运行当中是否可调用：是

参数：

Board_NO : 0－3, 板号

例（C 语言）：

```
long LineNO;
LineNO = LM_GetLineNO_9010 (0);
```

返回码：

当前行号; 范围: 长整型数; 用户设定值。当返回值=-2147483648 时,表示不成功,有错误产生。

参考函数：

§ 3.12.3.10 LM_GetFactVel_9010

获得实际插补速度

函数编号:1002

函数原型：

C: float APIENTRY LM_GetFactVel_9010(unsigned short Board_NO);

解释：

调用该函数将获得 9010 卡实际的插补速度。

轴正在运行时是否可调用：是

参数：

Board_NO : 0－3, 板号

返回码：

实际插补速度 单位: 用户单位/分钟

参考函数：

§3.12.3.11 LM_SetPauseTime_9010

设置插补暂停停止时间

函数编号:125

函数原型：

C： short APIENTRY LM_SetPauseTime_9010(unsigned short Board_NO,short Time);

解释：

这个函数设置插补暂停停止时间，执行函数 LM_Pause_9010 时，插补运动轴将在设定时间内停止。

在插补引擎正在运行当中是否可调用：是

参数：

Board_NO ： 0－3， 板号

Time ： 30-200； 暂停停止时间；单位：毫秒；缺省值： 30

返回值:

(0) 成功

(-1) 不成功

参考函数：

LM_Pause_9010

§3.12.3.12 LM_Pause_9010

暂停插补运算

函数编号:50

函数原型：

C： short APIENTRY LM_Pause_9010(unsigned short Board_NO);

解释：

这个函数将暂停插补运算，插补各轴在设定时间之内减速停止。
可用 LM_Resume_9010(short Board_NO)函数恢复。

在插补引擎正在运行当中是否可调用：是

参数：
Board_NO : 0－3, 板号

返回码：

(0) 成功
(-1) 不成功

参考函数：
LM_Resume_9010

§ 3.12.3.13 LM_Resume_9010

恢复插补运算
函数编号:51
函数原型：

C： short APIENTRY LM_Resume_9010(unsigned short Board_NO);

解释：

这个函数将恢复被函数（LM_Pause_9010）暂停的插补运算。

在插补引擎正在运行当中是否可调用：是

参数：
Board_NO : 0－3, 板号

返回值:

(0)　　成功

(-1)　　不成功

参考函数:

LM_Pause_9010

§3.12.3.14　LM_SetSpeedRate_9010

改变插补矢量速度

函数编号:52

函数原型:

C:　　short APIENTRY LM_SetSpeedRate_9010(unsigned short Board_NO,short Rate);

解释:

这个函数将实时改变插补矢量速度。设:用户设定速度=Vu;则:插补矢量速度 $=\frac{V_u \times Rate}{100.0}$

在插补引擎正在运行当中是否可调用:是

参数:

Board_NO :　0－3,　板号

Rate　　:　1-160, 100=100%,即:按原设定速度执行;

10=10%,即:按原设定速度的百分之十执行,

返回码:

(0)　　成功

(-1)　　不成功

参考函数:

§ 3.12.3.15 LM_LineEnd_9010

插补运行完当前行停止

函数编号:53

函数原型：

C： short APIENTRY LM_LineEnd_9010(unsigned short Board_NO);

解释：

插补运行完当前行停止。该命令不进入插补缓存区,可在插补运行时,随时执行 当插补引擎运行完当前行停止时,如要重新开始,则再执行 LM_Start_9010 命令

在插补引擎正在运行当中是否可调用：是

参数：

Board_NO : 0－3, 板号

返回码：

(0) 成功

(-1) 不成功

参考函数：

§ 3.12.3.16 LM_GetState_9010

获得当前插补状态

函数编号:56

函数原型：

C： short APIENTRY LM_GetState_9010(unsigned short Board_NO);

解释：

这个函数将获得当前插补引擎的状态。

在插补引擎正在运行当中是否可调用：是

参数：

Board_NO : 0－3, 板号

返回码：

0: 停止状态
1: 被轴停止命令结束 (建议用 Abort_9010 命令)
2: 被行结束停止命令停止
3: 被 LM_End_9010 结束
4: 插补缓冲区空 或 等待后续命令
5: 被进给倍率为 0 暂停
6: 被进给暂停挂起
7: 插补正在进行
255: 命令不成功

注意：当返回码为：4 或 5、6、7 时就是 **插补引擎正在运行** 状态。

参考函数：

§3.12.3.17 LM_SetSysPara_9010

设插补系统参数

函数编号:59

函数原型：

C： short APIENTRY LM_SetSysPara_9010(unsigned short Board_NO,double MinLength,double MinSpeed,double ArcError);

解释：

设置插补运算中的最小长度、速度和圆弧误差值。

在插补引擎正在运行当中是否可调用：否

参数：

Board_NO : 0－3, 板号
MinLength: 系统最小长度,单位: 用户单位, 缺省值: 0.001;在直线或圆弧插补时,直线或圆弧长度不能小于该设定值

MinSpeed : 系统最小速度,单位: 用户单位/分钟, 缺省值: 0.001;在直线或圆弧插补时,直线或圆弧插补速度不能小于该设定值

ArcError : 系统最大圆弧误差,单位: 用户单位, 缺省值: 0.2;在圆弧插补时,圆弧起点和终点半径误差不能大于该设定值

返回码:

大于 0 成功

小于 0 操作失败

参考函数:

GetErrorNo_9010

§ 3.12.3.18 LM_SetAxisMaxErrLtd_9010

设轴最大插补位置误差限制

函数编号:87

函数原型:

C: short APIENTRY LM_SetAxisMaxErrLtd_9010(unsigned short Board_NO,short Axis_No,double ErrLid);

解释:

设轴最大插补位置误差限制。

在插补引擎正在运行当中是否可调用:否

参数:

Board_NO : 0－3, 板号

Axis_No : 1－4, 轴标志号;1=X 轴,2=Y 轴,3=Z 轴,4=W 轴

ErrLid : 该轴最大插补定位误差限制值。单位为用户单位。如果超过该值,系统将报警。

返回码:

大于 0 成功

小于 0 操作失败

参考函数:

GetErrorNo_9010

§3.12.3.19 LM_SetSpeedPri_9010

设置插补速度速度优先还是精度优先

函数编号 ： 58

函数原型：

C： short APIENTRY LM_SetSpeedPri_9010(unsigned short Board_NO,short PriorityFlag);

解释：

该函数确定插补运算的两种效果，速度优先意味着运行相对更平滑，而精度优先意味着控制相对更精确

在插补引擎正在运行当中是否可调用：否

参数：

Board_NO ： 0－3, 板号

PriorityFlag ： 0-1; 1:插补速度优先, 0:插补精度优先。 缺省值: 1 插补速度优先

返回码：

(0) 成功

(-1) 不成功

参考函数：

§3.12.3.20 LM_GetFEnState_9010

获得轴编码器跟随已达到目标状态

函数编号:78

函数原型：

C： short APIENTRY LM_GetFEnState_9010(unsigned short Board_NO);

解释：

该函数将获得轴编码器跟随已达到或超过目标状态（由插补命令 LM_LineFE_9010 产生）。

在插补引擎正在运行当中是否可调用：是

参数：
Board_NO : 0－3, 板号

返回码：

0: 跟随还没有达到目标
1: 跟随已达到目标
255: 命令不成功

注意：跟随已达到目标，如果有新的插补模式下的轴跟随编码器位置命令开始执行，跟随状态将自动转为 0 状态。
参考函数：LM_LineFE_9010

§3.12.4 Nurbs 曲线插补函数

§3.12.4.1 LM_SetNurbsScanMode_9010

设置 Nurbs 曲线插补预处理的扫描模式
函数编号:1000
函数原型：

C: short APIENTRY LM_SetNurbsScanMode_9010(unsigned short Board_NO,short Mode,double ScanSpeed,double ScanSpeedRate);

解释：

设置在 Nurbs 曲线插补预处理时，对曲线扫描的速度设定。在预处理时，9010 卡按一定速度对 Nurbs 曲线进行逐点曲率半径计算，以找到曲率半径的凹点位置，该点是速度敏感点，9010 卡视情况将对该点位置前后进行速度控制。

9010 初始化时,缺省值是：

Mode =1

ScanSpeedRate=100

轴正在运行时是否可调用：否

参数：

Board_NO:　卡号，可能值 0，1，2，3

Mode　:　Nurbs 曲线插补预先扫描速度模式: 0=给定速度扫描,1=按程序速度百分比扫描

ScanSpeed　:　给定的扫描速度

ScanSpeedRate　:　按程序速度百分比的比值: 范围: 1-100

返回码：

(0)　成功；　(-1)　不成功

§3.12.4.2　LM_SetNurbsVelCtrl_9010

设置 Nurbs 曲线插补时的速度控制

函数编号:1001

函数原型：

C:　short APIENTRY LM_SetNurbsVelCtrl_9010(unsigned short Board_NO,short BSErrEnable,double BSErrV,short RAccEnable,double RAccV);

解释：

具体含义参看本书：**§2.6 NURBS 曲线** 节内容。

轴正在运行时是否可调用：否

参数：

Board_NO　卡号，可能值 0，1，2，3

BSErrEnable　:　0=控制弦高误差无效,1=控制弦高误差有效。缺省值：1

BSErrV　:　最大弦高误差值,范围: 0.0001-10.0, 单位: 用户单位。缺省值：0.001

RAccEnable　:　0=控制法向加速度无效,1=控制法向加速度有效。缺省值：0

RAccV　:　最大法向加速度值,范围: 1-1000000, 单位: 用户单位/秒/秒。缺省值：200

返回码：

(0)　成功；　(-1)　不成功

参考函数：

§3.12.4.3 LM_SetNurbsAccDec_9010

设置 Nurbs 曲线插补加速度和减速度

函数编号:93

函数原型：

C: short APIENTRY LM_SetNurbsAccDec_9010(unsigned short Board_NO,double Nurbs_Acc,double Nurbs_Dec);

解释：

设置 Nurbs 曲线插补的插补加速度和减速度。

轴正在运行时是否可调用：否

参数：

Board_NO 卡号，可能值 0，1，2，3
Nurbs_Acc: 插补加速度，范围：1-10000，单位：用户单位 / 秒/ 秒。缺省值: 500
Nurbs_Dec: 插补减速度，范围：1-10000，单位：用户单位 / 秒/ 秒。缺省值: 500

返回码：

(0) 成功； (-1) 不成功

参考函数：

§3.12.4.4 LM_SetNurbsCompCoef_9010

设 Nurbs 曲线插补误差补偿系数

函数编号:97

函数原型：

C: short APIENTRY LM_SetNurbsCompCoef_9010(unsigned short Board_NO,double Coef);

解释：

通过该函数，将改变 Nurbs 曲线插补的动态精度，插补误差补偿系数越大，动态精度会越高，但运行的平顺性将降低。

轴正在运行时是否可调用：否

参数：

Board_NO : 0－3, 板号
Coef: 设 Nurbs 曲线插补误差补偿系数，范围：0.0-0.08，单位：无。缺省值: 0.05

返回码：

0 或-1,-1=不成功,0=成功

参考函数：

§3.12.4.5 LM_NurbsInit_9010

初始化 Nurbs 曲线

函数编号:90

函数原型：

C: short APIENTRY LM_NurbsInit_9010(unsigned short Board_NO,unsigned short _deg);

解释：

用户在用到 Nurbs 曲线插补时，每组 Nurbs 曲线的第一行必须调用该函数，对 Nurbs 曲线内部数据进行初始化。

轴正在运行时是否可调用：是

参数：

Board_NO ： 卡号，可能值 0，1，2，3

deg　　：　(保留参数,恒等于 3) 3=三次 Nurbs 曲线

返回码：

(0)　成功；　(-1)　不成功

参考函数：

§3.12.4.6　LM_NurbsData_9010

设 Nurbs 曲线插补数据

函数编号:91

函数原型：

C:　short APIENTRY LM_NurbsData_9010(unsigned short Board_NO,double xPos,double yPos,double zPos,double knot,double weight);

解释：

设置 Nurbs 曲线插补数据，参数的具体含义请参看本书第 2.6 节（NURBS 曲线）及附录 A 的有关内容。

轴正在运行时是否可调用：是

参数：

Board_NO ：　卡号，可能值 0，1，2，3

xPos　　: X 轴的控制点位置(绝对值),单位为用户单位。
yPos　　: y 轴的控制点位置(绝对值),单位为用户单位。
zPos　　: z 轴的控制点位置(绝对值),单位为用户单位。
knot　　: 节点值。
weight　: 权值。

返回码：

(0)　成功；　(-1)　不成功

参考函数：

§3.12.4.7 LM_Nurbs_9010

Nurbs 曲线插补,固件 3.0 版

函数编号:92

函数原型：

C: short APIENTRY LM_Nurbs_9010(unsigned short Board_NO,double knot1,double knot2,double knot3,double knot4,double Speed,long LineNO);

解释：

设置 Nurbs 曲线插补。当用户调用到该函数时，一组 Nurbs 曲线插补数据设定结束，9010 卡的动态连接库开始对该组 Nurbs 曲线插补数据进行预处理，一般有以下步骤：

①检查数据是否完整，检查 Nurbs 曲线起点是否连续。

②对 Nurbs 曲线进行逐点扫描。

③形成中间数据，中间数据分两部分，一部分在插补缓冲区中排队，另一部分在数据缓冲区排队，视情况传入 9010 卡中。

轴正在运行时是否可调用：是

参数：

Board_NO ： 卡号，可能值 0，1，2，3

knot1 ：节点值 1。

knot2 ：节点值 2。

knot3 ：节点值 3。

knot4 ：节点值 4。

Speed ：插补速度,单位: 用户单位/分钟

LineNO ：插补行号,用户自由设定

返回码：

(0) 成功； (-1) 不成功

§3.12.4.8 LM_Nurbs4Axis_9010

4 轴 Nurbs 曲线插补

函数编号:92

函数原型：

C: short APIENTRY LM_Nurbs4Axis_9010(unsigned short Board_NO,double knot1,double knot2,double knot3,double knot4,double wPos,double Speed,long LineNO);

解释：

该函数是 4 轴 Nurbs 曲线插补，它指明 9010 卡的第 4 轴与 X、Y、Z 轴数据组成的 Nurbs 曲线按长度比例同动，其它情况与 LM_Nurbs_9010 函数类似。第 4 轴（W 轴）的移动速度按如下公式计算：

W 轴速度 $V = \frac{\mathrm{dL}_1}{dL_2} \times F$

其中： dL_1 表示 W 轴移动长度，dL_2 表示 NURBS 曲线的长度，F 表示指令速度 Speed。

轴正在运行时是否可调用：是

参数：

Board_NO ： 卡号，可能值 0，1，2，3

knot1 ： 节点值 1。

knot2 ： 节点值 2。

knot3 ： 节点值 3。

knot4 ： 节点值 4。

wPos : w 轴插补的终点位置(绝对值),单位为用户单位。

Speed ： 插补速度,单位: 用户单位/分钟

LineNO ： 插补行号,用户自由设定

返回码：

(0) 成功； (-1) 不成功

§ 3.12.4.9 LM_SendNurbsData_9010

向 9010 卡转输 NURBS 曲线数据

函数编号:1013

函数原型：

C: short APIENTRY LM_SendNurbsData_9010(unsigned short Board_NO);

解释：

由于 Nurbs 曲线可以由多个控制点组成（最少 4 个点），当控制点很多时，组成 Nurbs

曲线的数据急剧膨胀。在 9010 卡中有数据缓冲区，该函数就是负责将滞留在 9010 卡动态连接库中的数据发送到 9010 卡中。当用户调用插补顺序执行函数 和 LM_GetBuffLen_9010 (….) 函数 时，9010 卡的动态连接库会隐含调用该函数。一般情况下，当用户用到 Nurbs 曲线插补时，可在定时函数中显式调用该函数。

轴正在运行时是否可调用：是

参数：

Board_NO ： 卡号，可能值 0，1，2，3

返回码：

(0) 成功； (-1) 不成功

参考函数：

§ 3.12.4.10 LM_GetNurbsExecPara_9010

读取 9010 卡 NURBS 曲线插补运行参数值

函数编号:1012

函数原型：

C: float APIENTRY LM_GetNubrsExecPara_9010(unsigned short Board_NO);

解释：

当 9010 卡正在执行 Nurbs 曲线插补时，调用该函数可获得 Nurbs 曲线当前插补的**参数变量** Up 值。**参数变量** Up 值的含义请参看本书第 2.6 节（NURBS 曲线）及附录 A 的有关内容。通过该值可计算出 X、Y、Z 轴的理论位置，及 Nurbs 曲线从起点到插补点的长度。

轴正在运行时是否可调用：是

参数：

Board_NO ： 卡号，可能值 0，1，2，3

返回码：

当前 9010 卡 NURBS 曲线插补运行参数值

参考函数：

§3.12.4.11 LM_GetNurbsInBuffLen_9010

获得 Nurbs 曲线滞留在插补缓冲区的数

函数编号:1012

函数原型：

C: short APIENTRY LM_GetNurbsInBuffLen_9010(unsigned short Board_NO);

解释：

通过调用该函数可获得还有几条未执行的（包括正在执行的）Nurbs 曲线滞留在插补缓冲区。

轴正在运行时是否可调用：是

参数：

Board_NO ： 卡号，可能值 0，1，2，3

返回码：

0-256, Nurbs 曲线在插补缓冲区的数

-1=不成功

参考函数：

§3.12.4.12 LM_GetNurbsErrorNo_9010

获得产生 Nurbs 曲线(插补)的错误号

函数编号:1010

函数原型：

C: short APIENTRY LM_GetNurbsErrorNo_9010(unsigned short Board_NO);

解释：

在调用有关 Nurbs 曲线插补函数时，如果产生错误，可通过调用该函数获得产生错误的原因。

轴正在运行时是否可调用：是

参数：

Board_NO : 0－3, 板号

返回码：

Nurbs 错误号:

1=内存不够
2=超出计算范围
3=计算逻辑错误
4=扫描速度过快
5=数据不完整
6=节点矢量的数据超过数学公式的定义
7=控制点数不能少于 4
8=参数变量超过范围
-1=不成功

参考函数：

§3.12.4.13 Set_NurbsInit_9010

初始化 Nurbs 曲线,为 Nurbs 曲线计算做准备

函数编号:1004

函数原型：

C: short APIENTRY Set_NurbsInit_9010(unsigned short Nurbs_NO,unsigned short _deg);

解释：

纯粹是为了计算方便，9010 卡的动态连接库中预设了 8 条 Nurbs 曲线，用户可用这 8 条

曲线进行有关计算，主要是轴理论位置计算、Nurbs 曲线长度计算。有关细节请参看本书第 2.6 节（NURBS 曲线）及附录 A 的有关内容。

轴正在运行时是否可调用：是

参数：

Nurbs_NO : 0－7, Nurbs 曲线号

deg : (保留参数,恒等于 3) 3=三次 Nurbs 曲线

返回码：

0 或-1,-1=不成功,0=成功

参考函数：

§ 3.12.4.14 Set_NurbsData_9010

设 Nurbs 曲线计算数据

函数编号:1005

函数原型：

C: short APIENTRY Set_NurbsData_9010(unsigned short Nurbs_NO,double xPos,double yPos,double zPos,double knot,double weight);

解释：

设置 9010 卡动态连接库中预设 Nurbs 曲线的数据。该函数仅为计算使用，对插补函数没有影响。

轴正在运行时是否可调用：是

参数：

Nurbs_NO : 0－7, Nurbs 曲线号

xPos : X 轴的控制点位置(绝对值),单位为用户单位。

yPos : Y 轴的控制点位置(绝对值),单位为用户单位。

zPos : Z 轴的控制点位置(绝对值),单位为用户单位。

knot : 节点值。

weight : 权值。

返回码：

0 或-1,-1=不成功,0=成功

参考函数：

§3.12.4.15 Set_NurbsEnd_9010

设 Nurbs 曲线数据结束

函数编号:1006

函数原型：

C: short APIENTRY Set_NurbsEnd_9010(unsigned short Nurbs_NO,double knot1,double knot2,double knot3,double knot4);

解释：

数据设置结束。该函数仅为计算使用，对插补函数没有影响。

轴正在运行时是否可调用：否

参数：

Nurbs_NO : 0－7, Nurbs 曲线号

knot1 : 节点值 1。

knot2 : 节点值 2。

knot3 : 节点值 3。

knot4 : 节点值 4。

返回码：

0 或-1,-1=不成功,0=成功

参考函数：

§3.12.4.16 Get_NurbsPos_9010

计算 Nurbs 曲线轴位置(型值点)坐标

函数编号:1007

函数原型:

C: short APIENTRY Get_NurbsPos_9010(unsigned short Nurbs_NO,double Up,double *xPos,double *yPos,double *zPos);

解释:

9010 卡的动态连接库中预设了 8 条 Nurbs 曲线，数据设置结束后，可通过**参数变量** Up 计算该 Nurbs 曲线任意点的坐标值，比如，Up=0，将获得 Nurbs 曲线起点的坐标值；

Up=控制点数-3，将获得 Nurbs 曲线终点的坐标值。

轴正在运行时是否可调用：是

参数:

Nurbs_NO : 0－7, Nurbs 曲线号

Up : Nurbs 曲线参数变量,范围: 0- (控制点数-3)。

xPos : X 轴位置指针。

yPos : Y 轴位置指针。

zPos : Z 轴位置指针。

返回码:

0 或-1,-1=不成功,0=成功

参考函数:

§3.12.4.17 Set_NurbsPosVB_9010

计算 Nurbs 曲线轴位置(型值点)坐标（VB 风格）

函数编号:1008

函数原型：

C: double APIENTRY Get_NurbsPosVB_9010(unsigned short Nurbs_NO,double Up,short xflag,short yflag,short zflag);

解释：

9010 卡的动态连接库中预设了 8 条 Nurbs 曲线，数据设置结束后，可通过**参数变量** Up 计算该 Nurbs 曲线任意点的坐标值，比如，Up=0，将获得 Nurbs 曲线起点的坐标值；

Up=控制点数-3，将获得 Nurbs 曲线终点的坐标值。

轴正在运行时是否可调用：是

参数：

Nurbs_NO : 0－7, Nurbs 曲线号

Up : Nurbs 曲线参数变量,范围: 0- (控制点数-3)。
xflag : 为 1 时,指示返回 X 轴位置。
yflag : 为 1 时,指示返回 Y 轴位置。
zflag : 为 1 时,指示返回 Z 轴位置。

xflag、yflag、zflag 不能同时为 1。

返回码：

Nurbs 曲线在参数变量为 Up 的轴位置

参考函数：

§ 3.12.4.18 Get_NurbsLen_9010

计算 Nurbs 曲线长度

函数编号:1009

函数原型：

C: double APIENTRY Get_NurbsLen_9010(unsigned short Nurbs_NO,double Up);

解释：

9010 卡的动态连接库中预设了 8 条 Nurbs 曲线，数据设置结束后，可通过**参数变量** Up 计算该 Nurbs 曲线从起点到任意点的曲线长度，比如， Up=控制点数-3，将获得 Nurbs 曲线全长。

轴正在运行时是否可调用：是

参数：

Nurbs_NO : 0－7, Nurbs 曲线号

Up : Nurbs 曲线参数变量,范围: 0- (控制点数-3)。

返回码：

Nurbs 曲线在参数变量为 Up 的长度

参考函数：

§ 3.12.4.19 Get_NurbsErrorNo_9010

获得产生 Nurbs 曲线(计算)的错误号

函数编号:1011

函数原型：

C: short APIENTRY Get_NurbsErrorNo_9010(unsigned short Nurbs_NO);

解释：

在进行有关 Nurbs 曲线计算时，如果产生错误，可通过调用该函数获得产生错误的原因。

轴正在运行时是否可调用：是

参数：

Nurbs_NO : 0－7, Nurbs 曲线号

返回码：

Nurbs 错误号:

1=内存不够

2=超出计算范围

3=计算逻辑错误

4=扫描速度过快

5=数据不完整

6=节点矢量的数据超过数学公式的定义

7=控制点数不能少于 4

8=参数变量超过范围

9=电机控制卡初始化不成功

-1=不成功

参考函数：

§ 3.12.4.20 LM_SetForceCtrl_9010

设置插补力控制模式

函数编号:94

函数原型：

C: short APIENTRY LM_SetForceCtrl_9010(unsigned short Board_NO,short ForceFlag);

解释：

（保留函数）。

轴正在运行时是否可调用：否

参数：

Board_NO : 0－3, 板号

ForceFlag : 0-1; 0:插补力控制无效,1=插补力控制有效

返回码：

0 或-1,-1=不成功,0=成功

参考函数：

§3.13 状态读取函数

§3.13.1 9010 卡有关版本信息读取

§3.13.1.1 ReadFirmwareVersion_9010

读取 9010 控制卡固件版本号

函数编号:28

函数原型：

C: short APIENTRY ReadFirmwareVersion_9010(unsigned short Board_NO);

解释：

该函数返回 9010 控制卡上的 DSP 固件程序版本。

轴正在运行时是否可调用：是

参数：

卡号，可能值 0，1，2，3

返回码：

(0) 不成功

(>1) 成功。比如: 10 即代表固件为 1.0 版本

§3.13.1.2 ReadDllVersion_9010

读取 9010 控制卡动态链接库版本号

函数编号:67

函数原型：

C: short APIENTRY ReadDllVersion_9010();

解释：

该函数返回 9010 控制卡动态链接库的版本号。

轴正在运行时是否可调用：是

参数：

（无）

返回码：

(0)　　不成功

(>1)　　成功。比如:　10 即代表动态链接库的版本为 1.0 版本

§3.13.1.3　GetDriverVersion_9010

读取 9010 卡的驱动版本号

函数编号:70

函数原型：

C:　　int APIENTRY GetDriverVersion_9010(unsigned short Board_NO);

解释：

该函数返回 9010 控制卡驱动的版本号。（不重要）

轴正在运行时是否可调用：是

参数：

Board_NO : 0－3, 板号

返回码：

(0)　　不成功

(>1)　　成功。比如:　10 即代表 9010 卡的硬件驱动版本为 1.0 版本

§3.13.2 获取错误号

获取系统错误信息

函数编号:100

函数原型：

C： short APIENTRY GetErrorNo_9010(unsigned short Board_NO,short CleanFlag,unsigned int *ErrorNo,unsigned int *FuncNo);

解释：

获得错误号；当调用函数的返回值 为"不成功"时，或电机卡不正常时，可调用该函数获得错误信息

在插补引擎正在运行当中是否可调用：是

参数：

Board_NO : 0－3, 板号

CleanFlag : 0 或 1; 0=不清除错误; 1=清除错误;

*ErrorNo : 返回错误编号; 指针类参数,可以为 NULL(空指针)，VB 用户可以直接设为 0。

*FuncNo : 返回产生错误的函数编号;指针类参数,可以为 NULL(空指针)，VB 用户可以直接设为 0。

返回码：

(0) 无错误

(大于 0) 有错误，错误编号。

* 错误编号表:

1 轴位置寄存器超限
2 轴在插补运动中
3 插补误差超限
4 轴在运动中
5 逻辑错误
6 插补轴压在限位上

7 内存不够
8 参数错误
9 插补缓存器满
10 与电机卡通讯失败
11 与电机卡通讯超时
12 插补数据小于系统最小速度
13 插补数据小于系统最小长度
14 插补数据大于系统圆弧半径误差
15 插补数据圆弧计算错误

16 设定值越界；

系统值范围:

轴位置: -2147483648 至 2147483647

插补向量长度: <1073741823 (直线长度和圆弧长度)

圆弧最大半径: <1048575

插补圆心位置: -2147483648 至 2147483647

17 9010 动态链接库版本与 9010 卡的固件版本不一致

18 有插补测量代码,后续不能再加入插补命令

19 轴在跟随运动

20 轴在停止过程

21 Nurbs 曲线计算所需内存不够

22 Nurbs 曲线超出计算范围

23 Nurbs 曲线计算逻辑错误

24 Nurbs 曲线扫描速度过快

25 Nurbs 曲线数据不完整

26 Nurbs 曲线的起点不连续

27 Nurbs 曲线的数据没有及时传入卡内

28 多块 9010 卡的固件版本不一致

29 9010 卡固件版本太低

31 接收命令错误

32 接收插补数据错误

33 接收 NURBS 数据错误

34 当轴是闭环模式时,不能单独操作位置编码器

35 轴编码器反馈位置寄存器超限

36 轴位置跟随误差超限

37 轴找一转脉冲失败

38 轴 Home 点无效,GoHome 失效

参考函数：

§3.14 CAN 总线扩展函数

有关 CAN 总线扩展也可参看本书第 2.7 节（CAN 总线扩展）有关内容，对 CAN 总线扩展卡编程时，要参看该扩展卡有关说明，获得如下信息：

1、卡板地址（ID 号）

2、 扩展卡型号代码（CardType 值）

3、 输入输出字节数

§3.14.1 RegCANExp_9010

登记 CAN 总线扩展卡

函数编号:2000

函数原型：

C: short APIENTRY RegCANExp_9010(unsigned short Board_NO,unsigned long ID,short CardType);

解释：

登记 CAN 总线扩展卡。

轴正在运行时是否可调用：是

参数：

Board_NO	:	0-3,板号
ID	:	CAN 总线扩展卡 ID 号
CardType	:	扩展卡型号，E1044 和 E1044_RELAY 型扩展卡型号代码：1

返回码：

0 或-1, -1=不成功, 0=成功

§3.14.2 EnableCANExp_9010

CAN 总线扩展卡使能(初始化邮箱)

函数编号:116

函数原型：

C: short APIENTRY EnableCANExp_9010(unsigned short Board_NO);

解释：

使能 CAN 总线扩展卡。

轴正在运行时是否可调用：是

参数：

Board_NO: 0-3,板号

返回码：

0 或-1, -1=不成功, 0=成功

§3.14.3 SendCANData_9010

CAN 总线扩展卡发送数据（邮箱发送消息）

函数编号:117

函数原型：

C: short APIENTRY SendCANData_9010(unsigned short Board_NO,unsigned long ID,short CardType,unsigned long D1234,unsigned long D5678);

解释：

该函数向 CAN 总线扩展卡发送数据。

轴正在运行时是否可调用：是

参数：

Board_NO	：	0-3,板号
ID	：	扩展卡的 ID 号
CardType	：	扩展卡型号
D1234	：	低 4 字节
D5678	：	高 4 字节

返回码：

0 或-1, -1=不成功, 0=成功

§3.14.4 ReadCANL_9010

读取 CAN 总线扩展卡数据 低 4 字节

函数编号:2001

函数原型：

C: long APIENTRY ReadCANL_9010(unsigned short Board_NO,unsigned long ID);

解释：

该函数读取 CAN 总线扩展卡的数据低 4 字节。

轴正在运行时是否可调用：是

参数：

Board_NO ： 0－3, 板号

ID ： 扩展卡的 ID 号

返回码：

0-7、8-15、16-23、24-31 位,分别对应第 1、2、3、4 字节

§3.14.5 ReadCANH_9010

读取 CAN 总线扩展卡数据 高 4 字节

函数编号:2001

函数原型：

C: long APIENTRY ReadCANH_9010(unsigned short Board_NO,unsigned long ID);

解释：

该函数读取 CAN 总线扩展卡的数据，高 4 字节。

轴正在运行时是否可调用：是

参数：

Board_NO : 0－3, 板号

ID　　：扩展卡的 ID 号

返回码：

0-7、8-15、16-23、24-31 位,分别对应第 5、6、7、8 字节

§3.14.6　GetCANErrorNo_9010

获得 CAN 总线错误号

函数编号:117

函数原型：

C:　　short APIENTRY GetCANErrorNo_9010(unsigned short Board_NO,short CleanFlag);

解释：

当调用有关 CAN 总线扩展卡的函数返回值 为"不成功"时，或电机卡不正常时,可调用该函数获得错误信息

轴正在运行时是否可调用：是

参数：

Board_NO　：　0－3, 板号

CleanFlag　：　0 或 1;　0=不清除错误;　1=清除错误;

* 返回值 :　0-9, 0=无错误,错误号

*

* 错误编号表:

1	邮箱号越界
2	数据长度越界
3	邮箱已经占用
4	邮箱的 ID 号重复
5	CAN 总线只能使能一次
6	邮箱不匹配
7	CAN 总线接受数据超过限制 (>64 字节)

8	邮箱已满
9	CAN 总线接收数据无效

返回码：

0 或-1, -1=不成功, 0=成功

附录 A　NURBS 曲线详解

1、NURBS 曲线定义及参数含义

NURBS（Non Uniform Rational B-spline）曲线通常称为非均匀有理 B 样条曲线，其数学定义如下：

，$P(K)$ 为曲线上的位置向量，$N_{i,m}(K)$为m次样条基函数

基函数由递推公式定义：

$$N_{i,0}(K)=\begin{cases}1(K_i \le K \le K_{i+1})\\0(\text{其它})\end{cases}$$

$$N_{i,m}(K)=\frac{(K-K_i)N_{i,m-1}(K)}{K_{i+m}-K_i}+\frac{(K_{i+m+1}-K)N_{i+1,m-1}(K)}{K_{i+m+1}-K_{i+1}},\quad m\ge 1$$

式中：P_i 为控制点；R_i 为权因子；K 为节点矢量

非均匀：指节点向量的值与间距可以为任意值。这样我们可以在不同区间上得到不同的混合函数形状，为自由控制曲线形状提供了更大自由。均匀与非均匀的主要区别在于节点向量的值。如果适当设定节点向量，可以生成一种开放均匀样条，它是均匀与非均匀的交叉部分。开放样条在两端的节点值会重复 d 次，其节点间距是均匀的。例如：

{0, 0, 1, 2, 3, 3}，（d=2，n=3）

{{0, 0, 0, 1, 2, 2, 2}，（d=4, n=4）。

开放均匀 B 样条与贝泽尔样条性质非常类似，如果 d=n+1（即多项式次数为 n），那么开放 B 样条就变成了贝泽尔样条，所有节点值为 0 或 1。如四个控制点的三次开放 B 样条，节点向量为：{0, 0, 0, 0, 1, 1, 1}。

有理 B 样条：有理函数是两个多项式之比，有理样条(rational spline)是两个样条函数之比，有理 B 样条用向量描述。

NURBS 曲线由以下三个参数定义：

（1）　控制点 P_i ：确定曲线的位置，通常不在曲线上，形成控制多边形。（见图 1,图中 $B_i=P_i$ ）

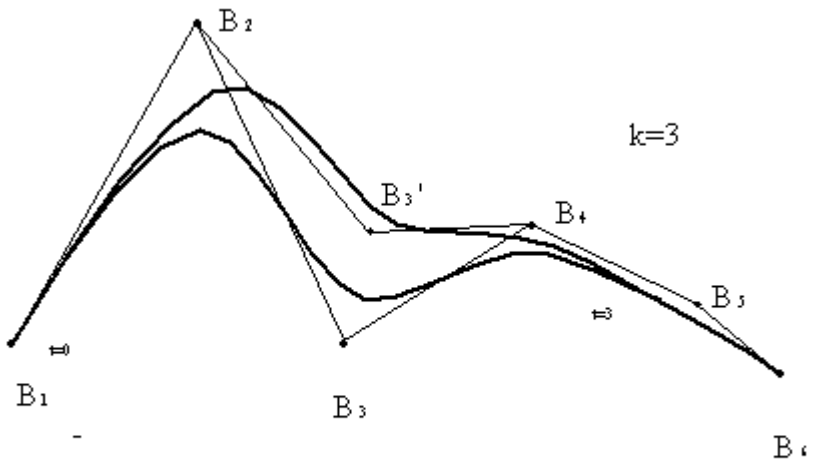


图1　控制点移动对曲线的影响

（2）　权因子 $R_i(R_i>0)$：确定控制点的权值，它相当于控制点的“引力”，其值越大曲线就越接近控制点（见图 2，B_i 为控制点）。

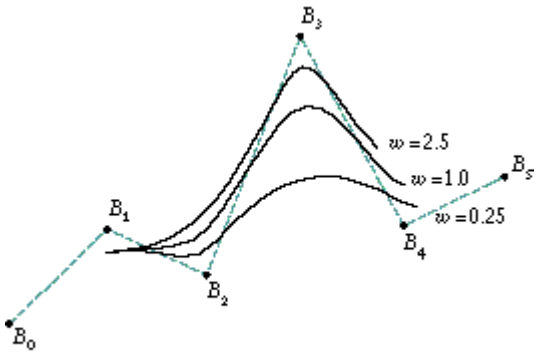


图 2 曲线随权因子变化

（3）　节点矢量 K：NURBS 曲线随着参数 K 的变化而变化，与控制顶点相对应的参数化点 K 称为节点，节点的集合 K_i：$[K_0, K_1\ldots, K_n\ldots, K_{n+m+1}]$称为节点矢量。
节点：在曲线上任意一点有多于一个控制点产生影响（除了 bezier 的端点），节点就象一种边界，在这个边界上一个控制点失去影响作用，另一个控制点取得影响。

2、NURBS曲线怎样通过首末节点

多重节点序列使得样条曲线更靠近于重复节点位置。如果末端节点重复 d+1 次，则 d 阶 B-样条必须插值最后一个控制点。因此，解决样条曲线不能横跨整个控制顶点序列的一个方法是，重复首尾两个节点，这样得到的样条曲线将插值首尾两个控制点。

对于非周期的样条，节点矢量为

$$U=\left[\underbrace{0,0,\cdots,0}_{m+1},u_{m+1},\cdots,u_{m-k-1},\underbrace{1,1,\cdots 1}_{m+1}\right]$$

即节点矢量两端各有 m+1 个相同的节点，以便曲线通过控制多边形首、末端点，并与首、末两边相切。

（参考系统编程篇§2.7程序示例）

3、NURBS曲线轨迹的矩阵计算法及矩阵表示

因NURBS样条函数的节点参数沿参数轴的分布是不等距的，不同节点矢量形成的B样条基函数各不相同，需要单独计算，且算法中又增加了权因子，所以曲线轨迹点的计算较为复杂、费时，为了提高NURBS曲线插补的实时性，在实时插补前需进行必要的预处理，其主要任务是确定NURBS曲线轨迹计算公式的有关系数，以简化实时插补的计算量。

若曲线采用三次NURBS形式表示（三次与K次计算方法相同，表达式不同），即K=3，$t\in[0,1]$，则第i段曲线可以写成下列矩阵形式：

$$P_i(t)=\frac{\sum_{j=i-3}^{i}w_i d_i N_{i,3}(u)}{\sum_{j=i-3}^{i}w_i N_{i,3}(u)}=\frac{\left[1,t,t^2,t^3\right]M_i\begin{bmatrix}w_{i-3}d_{i-3}\\w_{i-2}d_{i-2}\\w_{i-1}d_{i-1}\\w_i d_i\end{bmatrix}}{\left[1,t,t^2,t^3\right]M_i\begin{bmatrix}w_{i-3}\\w_{i-2}\\w_{i-1}\\w_i\end{bmatrix}}\qquad ，\ t\in[0,1]，\ i=3,4,\cdots,n$$

整理可得：

$$P_i(t)=\frac{C_0+C_1t+C_2t^2+C_3t^3}{C_0^{'}+C_1^{'}t+C_2^{'}t^2+C_3^{'}t^3}\qquad ，\ t\in[0,1]，\ i=3,4,\cdots,n$$

由于控制点d_i及权因子w_i均已知，而M_i仅与节点向量有关，也是确定的，C_i与M_i、w_i、d_i有关，即也是确定的，故式中各项系数均已知，且与插补点的参数无关，可在插补前一次性求出，因式中i的取值为 3 到n，所以对整条NURBS曲线，可计算出的系数共有$n-2$组，在插补中根据插补点所在的位置动态选用相应的系数。

对于曲线上坐标X、Y、Z分别有：

$$P_x(t)=\frac{C_{Ax_1}+C_{Ax_2}t+C_{Ax_3}t^2+C_{Ax_4}t^3}{C_{Ax_1}^{'}+C_{Ax_2}^{'}t+C_{Ax_3}^{'}t^2+C_{Ax_4}^{'}t^3}，\quad P_y(t)=\frac{C_{Ay_1}+C_{Ay_2}t+C_{Ay_3}t^2+C_{Ay_4}t^3}{C_{Ay_1}^{'}+C_{Ay_2}^{'}t+C_{Ay_3}^{'}t^2+C_{Ay_4}^{'}t^3}$$

，　$P_z(t)=\dfrac{C_{Az_1}+C_{Az_2}t+C_{Az_3}t^2+C_{Az_4}t^3}{C'_{Az_1}+C'_{Az_2}t+C'_{Az_3}t^2+C'_{Az_4}t^3}$　　，　$t\in[0,1]$，　$i=3,4,\cdots,n$