

Turbo PMAC 培训

培训者:

Stephen Milici
Richard Naddaf
Sina Sattari

课程综述

- n 简介
- n 软件包
- n 操作入门
- n 门阵列介绍
- n PMAC的多任务处理
- n PMAC的变量
- n PMAC的参数设置
- n PMAC的手动命令
- n 触发式手动命令 (Homing)
- n 伺服环的整定
- n 控制器 / 伺服器接口的发展
- n PMAC的换相
- n I²T 保护
- n PMAC的补偿表
- n PMAC 1/T扩展
- n PMAC的坐标系
- n PMAC的位置跟踪 (主从方式)
- n 运动程序
- n 线性运动模式
- n PMAC的数据采集
- n 圆弧运动模式
- n PMAC的前瞻技术
- n 快速运动模式
- n 样条曲线运动模式
- n PVT运动模式
- n 子程序
- n PMAC的PLC程序
- n PMAC的内存映射和DPRAM
- n PMAC的伺服控制算法
- n PMAC的时基控制
- n PMAC的维护

简介

PMAC及其各种版本的介绍

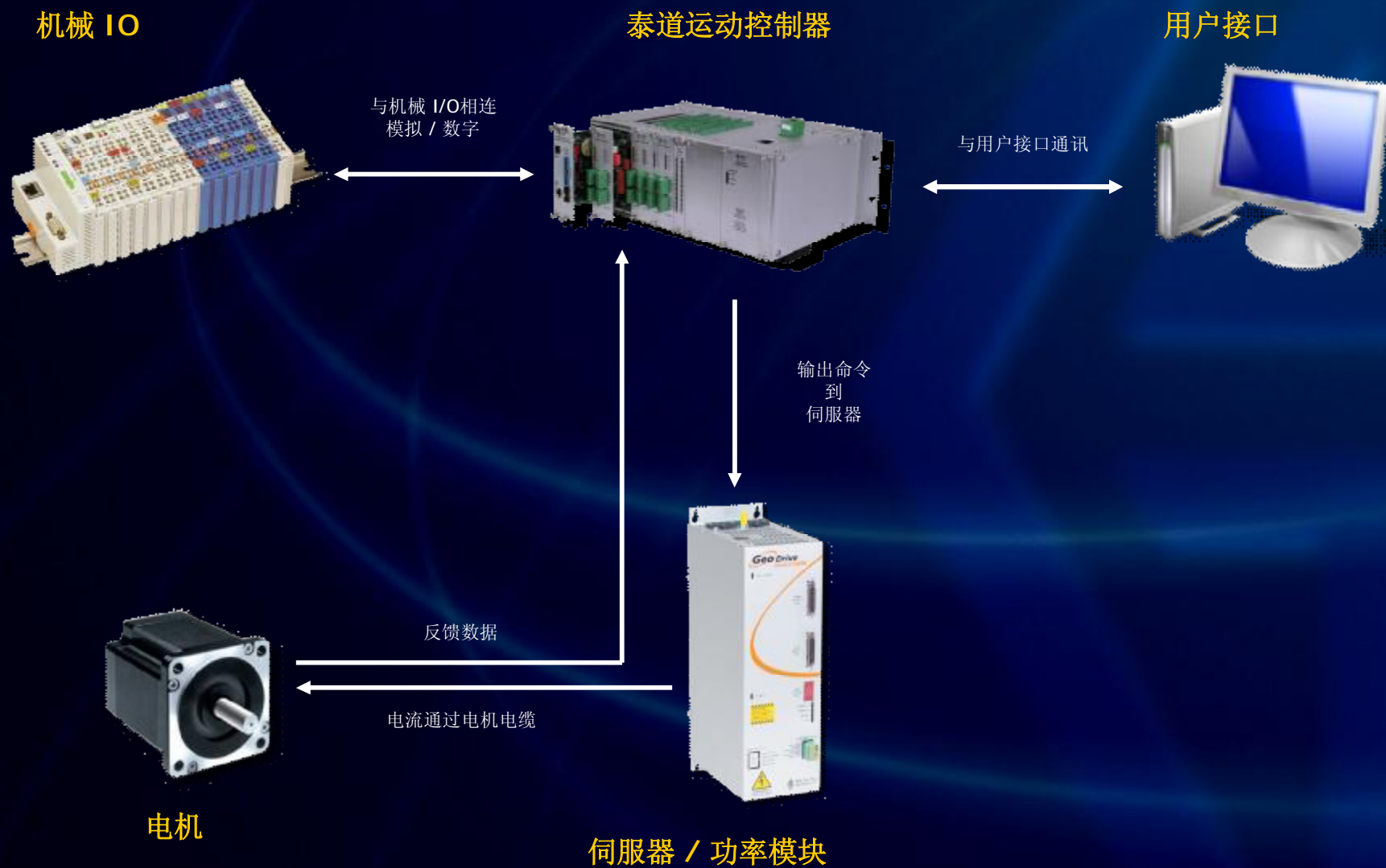
PMAC概况

PMAC的意思是可编程多轴运动控制器（**P**rogrammable **M**ulti-**A**xis **C**ontroller）

从次微米精度加工到需要几百千瓦或马力的大功率驱动，PMAC在广泛的领域都能够应用。

- n 机器人
- n 食品加工
- n 机床
- n 印刷
- n 造纸加工
- n 包装过程
- n 装配线
- n 物料搬运
- n 相机控制
- n 硅片加工
- n 自动焊接
- n 激光切割

整个系统原理图



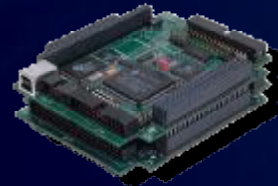
不同类型产品



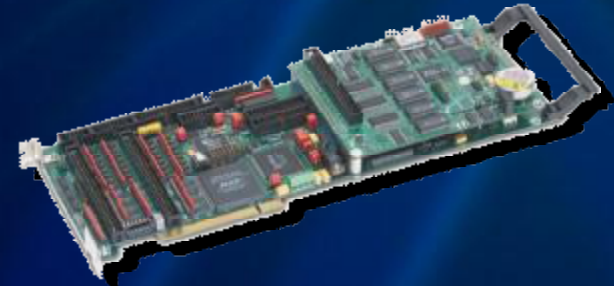
总线产品



VME



PC104



PCI

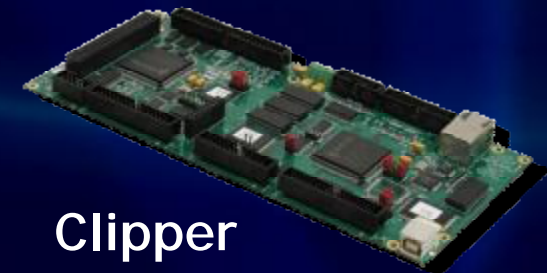
独立系统产品 USB / Ethernet



Brick Products



UMAC



Clipper

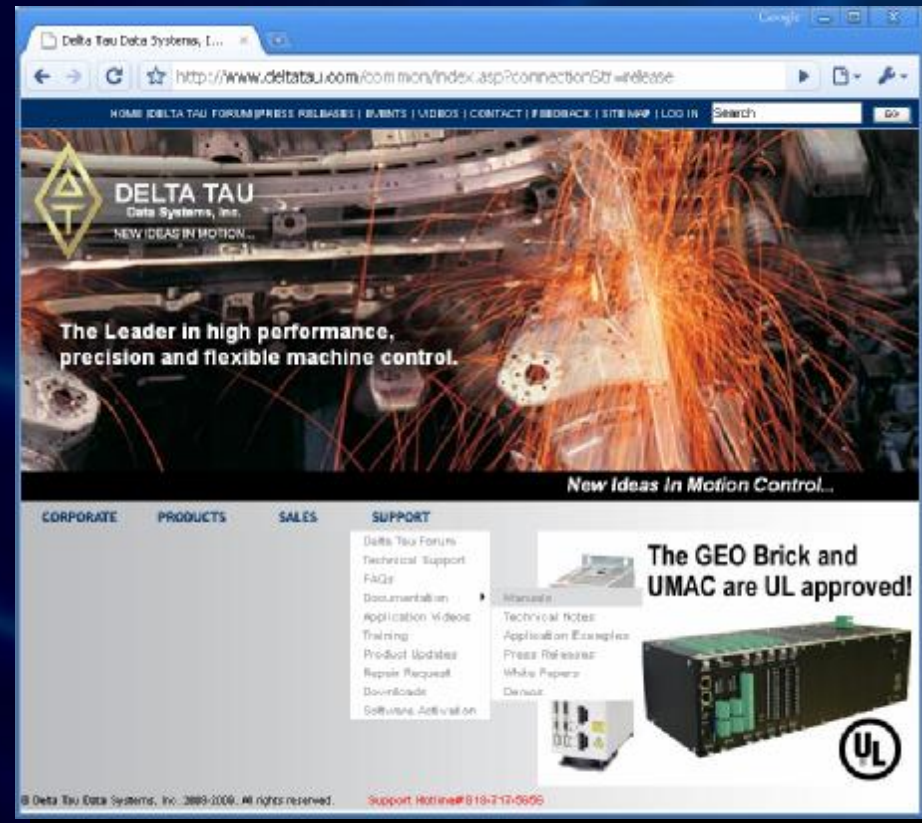
不同的PMAC卡及其特征

	独立控制器运行	多端口同时换相	与PC机的通讯接口				板卡上 DPRAM可选	板卡上的 ADC	板卡上的 16/32 位 I/O	输出命令信号			
			PCI	USB	Ethernet	Serial				模拟输出 ($\pm 10V$)	脉冲宽度调制 (PWM)	脉冲频率调制 (PFM)	光纤接口MACRO
PMAC	●		●			●	○			●			
PMAC2	●		●	●	●	●	●	●	●	●	●	●	●
Turbo PMAC	●	●	●			●	●		●	●			
Turbo PMAC2	●	●	●	●	●	●	●	●	●	●	●	●	●

信息资源

网站: www.deltatau.com

- n 产品信息
- n 分销商和厂商的联系信息
- n 最新的手册
- n 技术笔记
- n 白皮书
- n 软件下载



软件包

软件包



PEWIN32-Pro2 Suite (开发环境)

- n 主要的通讯和设置
- n PMAC Tuning Pro2 (偏差校正, PID环和电流环...)
- n PMAC Plot Pro2
- n 专门的硬件应用设置

通讯库

- n 32位通讯驱动 (与64位兼容)
- n 与Borland和Microsoft .NET开发的工具兼容

PMAC NC PRO2

- n 为PC上实现CNC控制的基于Windows定制用户界面 (GUI)

PMAC HMI

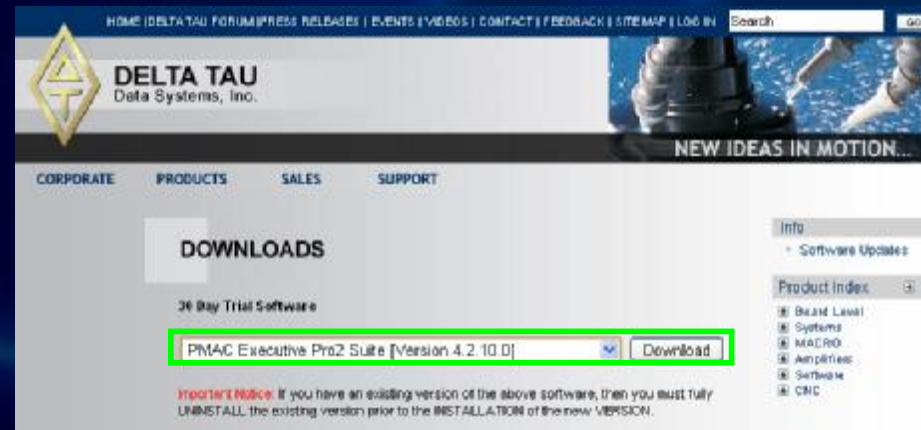
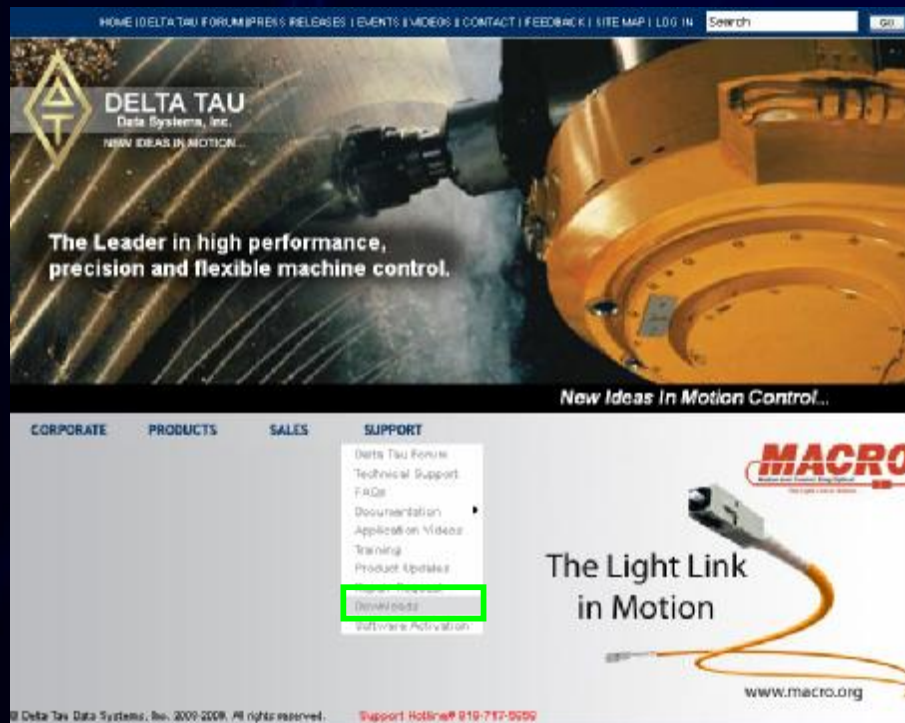
- n 供前端开发应用的强大易用的软件包

PMAC Executive PRO2 Suite

- n PMAC Executive Pro2
 - n 主要的通讯和设置工具
- n PMAC Tuning Pro2
 - n DAC偏差校正, PID环和电流环调节
- n PMAC Plot Pro2
 - n 允许用户从任意PMAC内存和/或I/O地址采集信息并以不同的形式绘图.
- n 设置程序 (P1, P2, Turbo Setup Pro2, Geo Brick Setup, 等.)
 - n 用于帮助设置不同类型PMAC卡、驱动器和电机.
- n UMAC Configure Pro2
 - n 用于配置UMAC

哪里可以得到最新版本的 PMAC Executive PRO2 Suite?

总可以在泰道公司的官方网站下载到最新发布的PMAC Executive PRO2。



PMAC Executive PRO2 Suite的安装

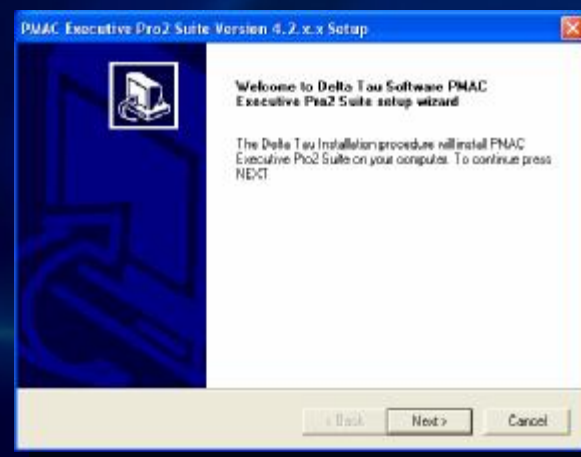
n 下载后，将ZIP文件解压到一个文件夹中。

n 执行 Setup.exe (需要有管理权限)

n 根据提示指令完成安装。

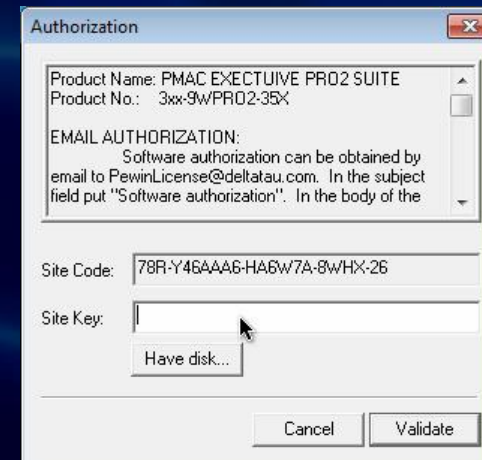
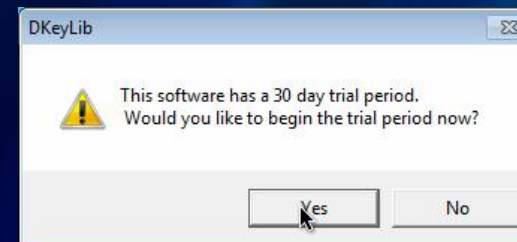
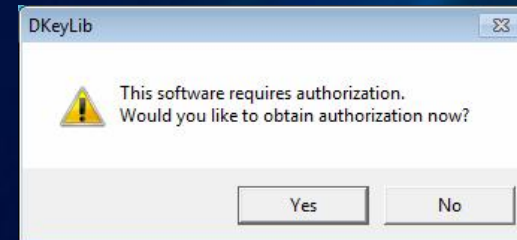
n 桌面和开始菜单中将会添加多个图标。

n 运行 Pwin32PRO2 软件。



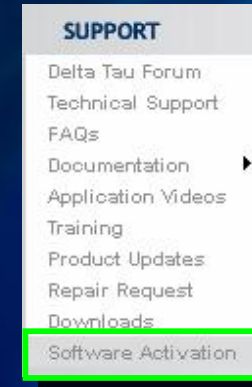
PEWIN32PRO2的授权

- n 当第一次运行PEWIN32PRO2 软件时，你将被告知需要授权使用。
- n 如果你不想提供授权的话，将得到的是30天的试用版。过期后，软件将不能使用，你必须提供授权或者重新安装，方可使用。
- n 如果你选择提供软件授权，则会出现授权窗口，显示PC自动产生的Site Code。



PEWIN32PRO2的授权 (续)

- n 为了得到软件授权，你需要到泰道的官方网站获得 **Site Key** 。
- n 在网站上，从**Support**菜单中选择 **Software Activation**。
- n 填写好表格中的必填项。
- n 泰道公司的销售发票和你的软件CD上都有你的产品序列号信息。
- n 填写好这页信息后，下一步将会要求输入你的**Site Code**，然后系统会为你产生**Site Key** 。

A screenshot of a web form titled 'Customer Information'. It contains several input fields for user registration: Name, Email Address, Company, Address1, Address2, City, State, Zip, Country, Phone Number, Fax, Serial Number, and Customer P.O. Each field is preceded by a red asterisk (*). At the bottom right of the form is a 'Continue' button.

建立与控制器的通讯

n 两种通讯端口

n 即插即用型端口

- n PCI

- n USB

n 非即插即用型端口

- n ISA (Legacy)

- n 串口Serial

- n 以太网Ethernet

n 即插即用型端口将会自动监测并为它们安装好驱动程序。

n PMAC Executive PRO2 Suite中都是最新的驱动程序。

n 非即插即用型端口需要手动安装驱动。

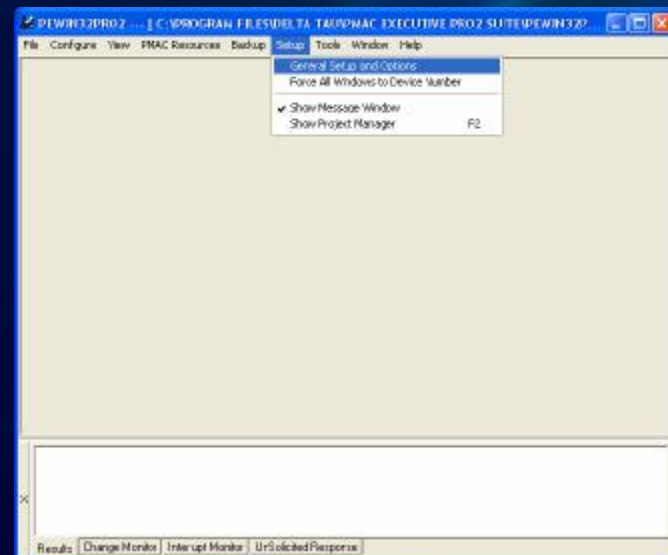
- n ISA需要泰道公司提供驱动程序。

- n Serial端口的驱动程序可能在Windows系统中，或者由端口硬件生产厂商提供。

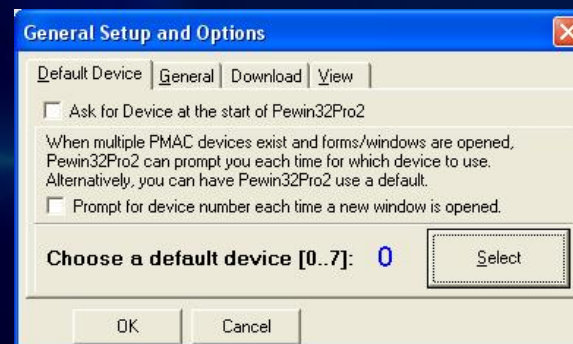
- n Ethernet不需要驱动， windows下已经有Ethernet的硬件。

建立与控制器的通讯(续)

- n 在PEWIN32PRO2中，从Setup菜单下选择General Setup and Options项。

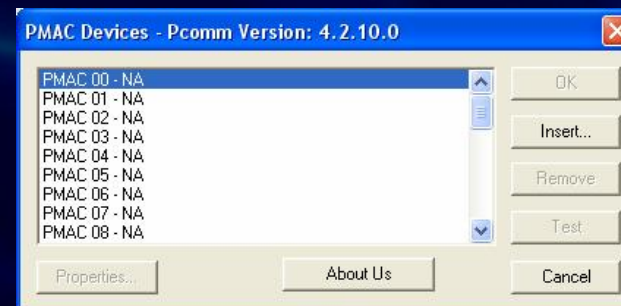


- n 在General Setup and Options窗口，点击Select按钮。



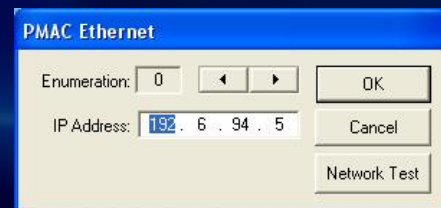
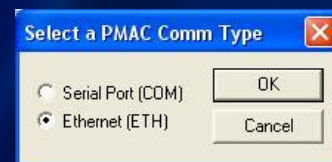
- n 第一次使用时，系统中没有选定用来通讯的PMAC卡。

- n 点击Insert ... 按钮。



建立与控制器的通讯(续)

- n 点击**Insert**按钮后，将会显示还没被PEWIN32PRO2使用的所有可用 PMAC 卡的列表。
- n 列表将只自动显示即插即用型设备。
- n 对于非即插即用型通讯端口 (Serial & Ethernet)，可点击**New**按钮，然后添加serial端口或者新的Ethernet IP地址。
- n 对于Ethernet通讯须注意，要么PC机在同一子网中，要么需在Windows的路由列表中添加**静态路由**。对任何基于Ethernet的PMAC，这些可以通过点击**Test**按钮来实现。



建立与控制器的通讯(续)

- n 建议在建立PMAC与PEWIN32PRO2通讯前进行测试。
- n 一旦通讯成功，即可开始运行PEWIN32PRO2。

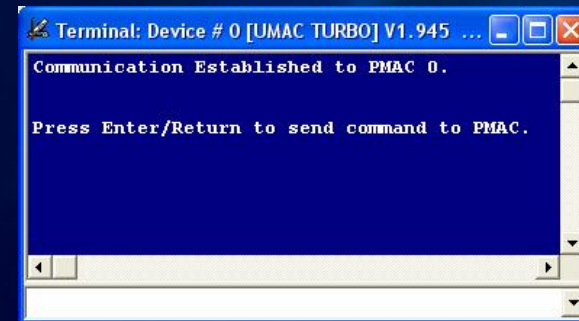


PEWIN32PRO2的特点

PEWIN32PRO2下的View菜单可以得到以下所有窗口。

n 终端窗口(Terminal Window)

- n 在线编辑和发送命令
- n 查询PMAC变量



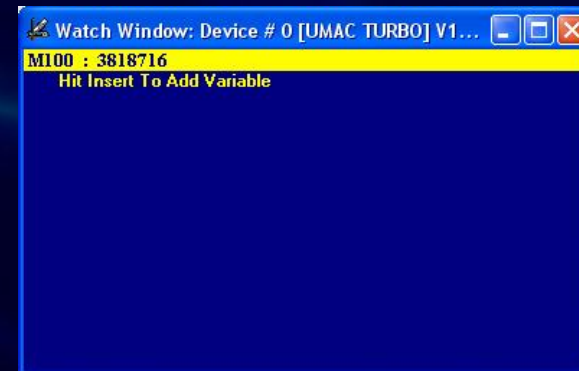
n 位置窗口(Position Window)

- n 显示位置、速度和跟随误差
- n 单位换算

	Position	Velocity	Fol. Error
# 1:	0.0 ct	0.0 ct/sec	0.0 ct
# 2:	0.0 ct	0.0 ct/sec	0.0 ct
# 3:	0.0 ct	0.0 ct/sec	0.0 ct
# 4:	0.0 ct	0.0 ct/sec	0.0 ct
# 5:	0.0 ct	0.0 ct/sec	0.0 ct
# 6:	0.0 ct	0.0 ct/sec	0.0 ct
# 7:	0.0 ct	0.0 ct/sec	0.0 ct
# 8:	0.0 ct	0.0 ct/sec	0.0 ct

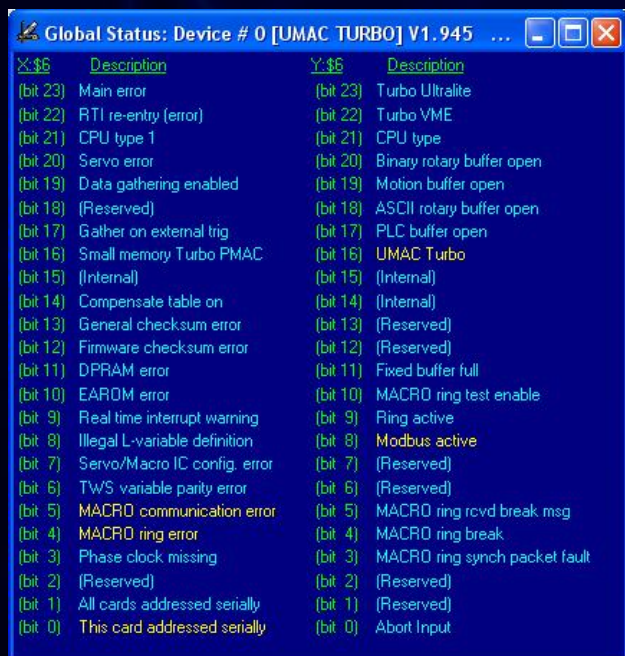
n 观察窗口(Watch Window)

- n 实时动态显示PMAC变量
- n 可直接读取内存内容
- n 错误命令将导致潜在危险



PEWIN32PRO2的特点(续)

- n 全局状态窗口(Global Status Window)
 - n 显示控制器状态
 - n 显示将会导致PMAC任务失败的全局错误报告

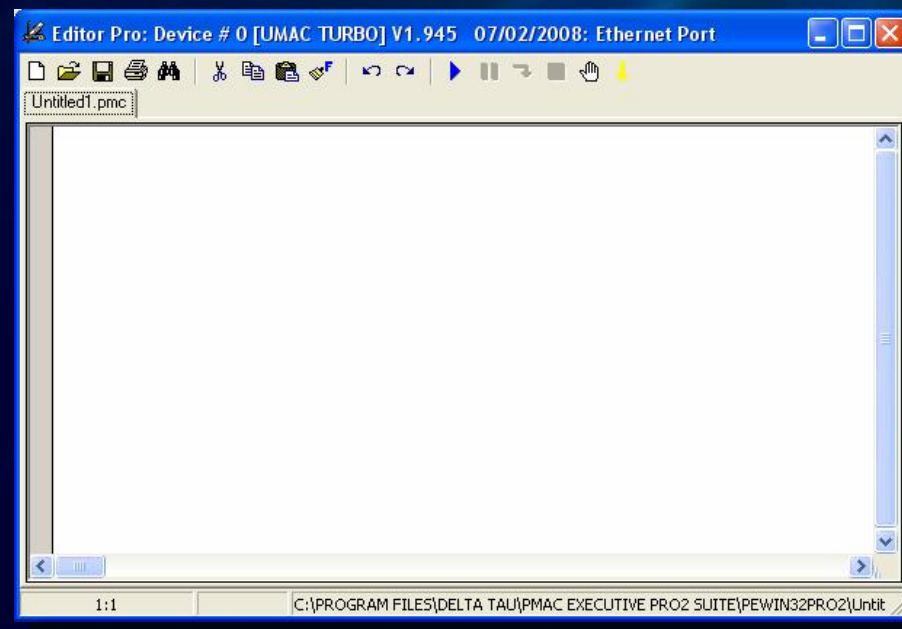


- n 电机状态窗口
 - n 用于得到每个电机的状态
 - n 用于故障诊断
 - n 用于跟踪程序的运行过程

PEWIN32PRO2的特性(续)

n 编辑窗口(Editor Window)

- n 编写运动程序
- n 编写PLC程序
- n 下载文件
- n 调试代码
- n 显示一些控制器查询结果



PEWIN32PRO2的特点(续)

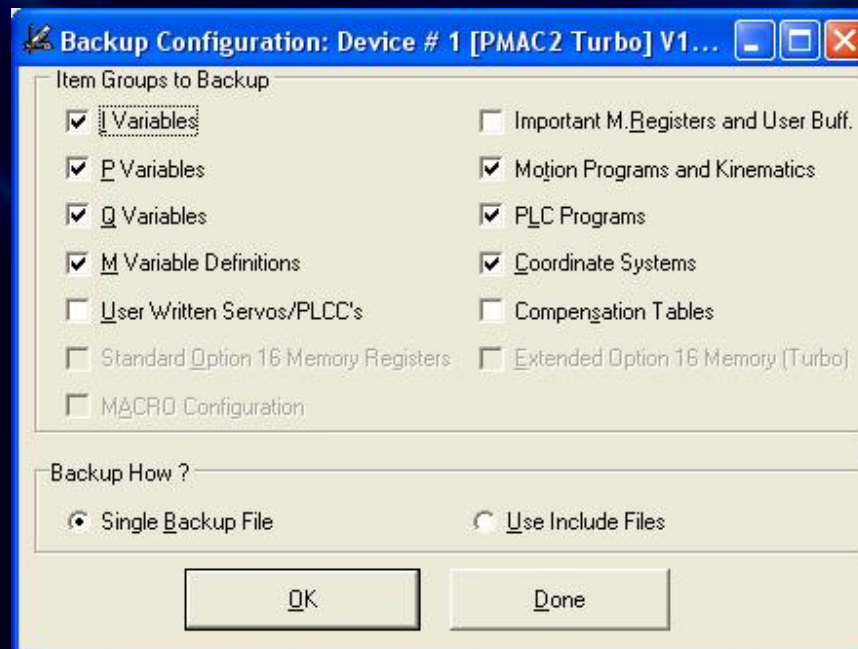
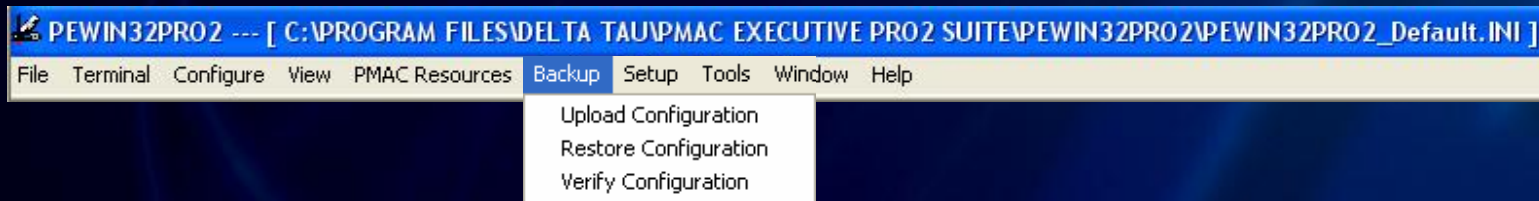
- n 备份菜单(Backup Menu)
 - n 将所有的配置设定备份到一个文件中
 - n 将保存的配置文件复原到PMAC中
 - n 对PMAC内存内容和配置文件进行比较/验证



操作入门

注意！！

- n 将PMAC内存内容备份并存入计算机。



使电机转起来！

- n 尝试在终端窗口（PEWIN32PRO2>View Menu>Terminal）输入以下命令。在终端窗口输入每条命令后，敲击键盘“回车”键，将命令发送到PMAC。

- n #1J+

- n #1J-

- n #1O20 (字母O后是数字 20)

- n #1O-20 (字母O后是数字 -20)

- n #1K

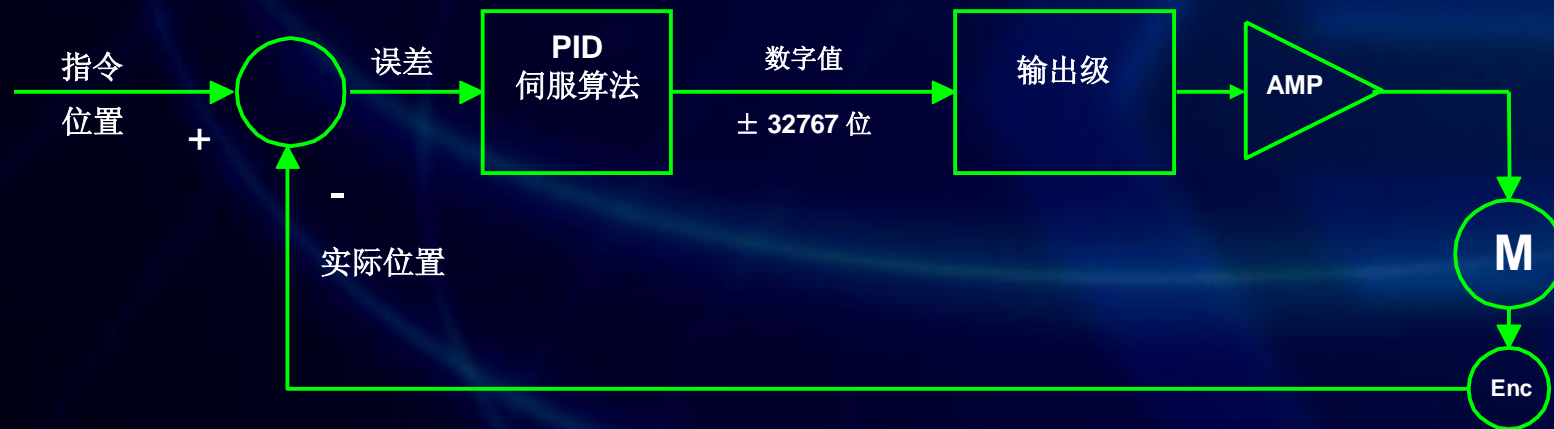
- n 现在可以观察电机1的运行结果
- n 命令#1j+ 和 #1O20有何不同
- n 命令“K”有何作用？

命令J 和 O的不同

- n 前面我们命令电机完成两种不同的运动
 - n J 为手控闭环运动(Jog): 命令电机在可控的闭环算法下运行。电机的位置、速度和加速度都由PMAC控制。
 - n O 为开环(Open loop)运动: PMAC只对驱动输出转矩或者速度命令。在这种情况下, 只有驱动/电机的机械和电子特性来限制/控制电机的位置/速度/加速度。
- n 显然如果需要控制电机的位置, 闭环控制将更能有效的控制电机(机器)

什么是闭环控制？

- n 闭环控制——通过实时比较电机的实际位置和指令位置，以得到正确的控制输出量。



再让电机转起来！

- n 在终端窗口输入以下命令
 - n I122
- n 输入命令 “#1j+” 使电机正方向手动运行
- n 输入以下命令为变量I122分配新的值
 - n I122=64
- n 再输入一次 “#1j+”
- n 观察这时电机的运行有什么不同？

记住输入 “#1K” 以停止电机运动。

I122的用处?

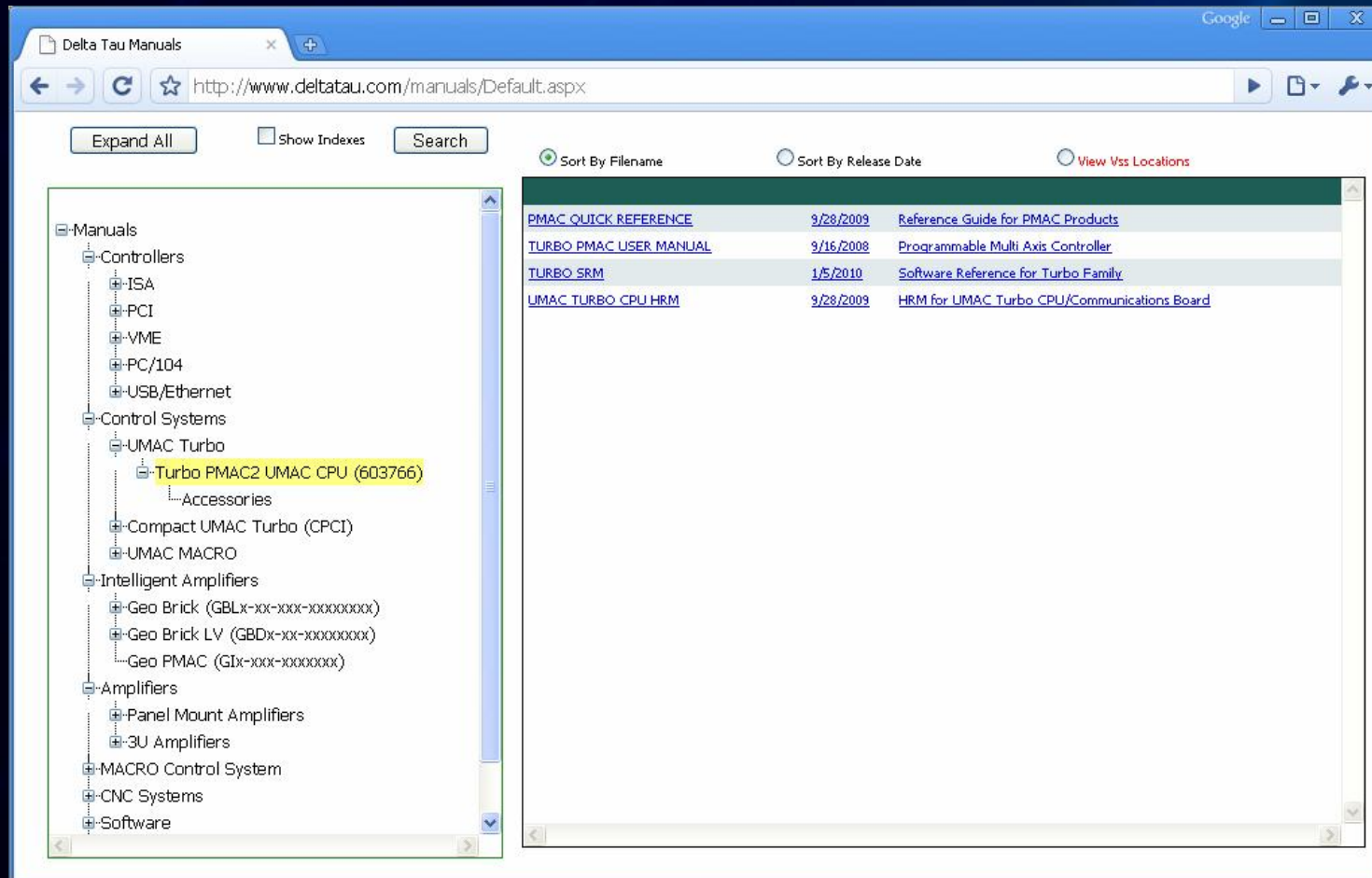
- n 注意到上一个测试结果:
 - n I122变量可以存储一个值，并可以按要求读取出来。
 - n 使用 "=" 号为I122变量赋值。
 - n 改变I122变量，将影响对应电机的运行速度。
 - n 只有当再一次运行"#1j+"命令时，电机运动的速度才会改变。
- n 因此刚才对I122赋了新值，结果将怎样?
- n 什么是 I122?
- n 字母 I 代表什么?

哪里可以找到 I122 的定义?

- n 有几个地方可以找到 I122 的定义和一些相关文档
 - n 泰道公司的官方网站 (www.deltatau.com)
 - n 产品信息
 - n 销售商/分销商信息
 - n 泰道公司手册网站 (www.deltatau.com/manuals)
 - n 根据不同产品的版本和系列进行了归类
 - n 软件参考手册
 - n 硬件参考手册
 - n 泰道公司论坛 (forums.deltatau.com)
 - n FAQ(经常问到的问题)
 - n 应用笔记
 - n Q&A

软件参考手册实例

- n 在手册网站找到Turbo软件参考手册 (SRM) :



软件参考手册实例

- n 在Turbo SRM中查找Ixx22. xx表示第几个电机。

Ixx22 Motor xx Jog Speed

Range: Positive Floating Point

Units: counts / msec

Default: 32.0

Ixx22 establishes the commanded speed of a jog move, or a programmed **RAPID**-mode move (if Ixx90=0) for Motor xx. Direction of the jog move is controlled by the jog command.

A change in this parameter will not take effect until the next move command. For instance, to change the jog speed on the fly, start the jog move, change this parameter, then issue a new jog command.

PMAC命令和 如何使用参考手册

三个地方寻找答案:

n Turbo PMAC软件参考手册 (Turbo SRM)

- n 所有Turbo PMAC的变量、命令和寄存器的详细信息
- n 变量和寄存器按数字顺序进行一一说明
- n 命令按字母顺序进行说明。

n Turbo PMAC用户手册

- n 设置Turbo PMAC应用的操作指南
- n 系统整体配置
- n 单个电机的设置
- n 如何使用坐标系/通道
- n 编写运动程序
- n PLC程序
- n 主机通讯程序

n 硬件参考手册（针对控制器/附件）

- n 硬件配置
- n 软件配置
- n 跳接设定
- n 插脚引线

门阵列介绍

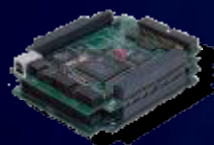
PMAC 硬件

PMAC是一种非常灵活的控制器，适用于不同类型的主机、伺服器、电机和传感器，以及不同领域的应用。

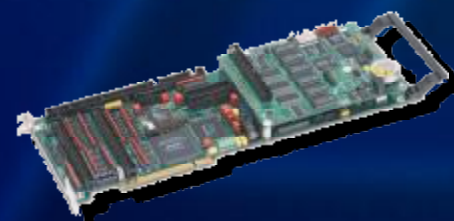
特定总线产品



VME



PC104



PCI

独立产品
USB / Ethernet



Brick Products

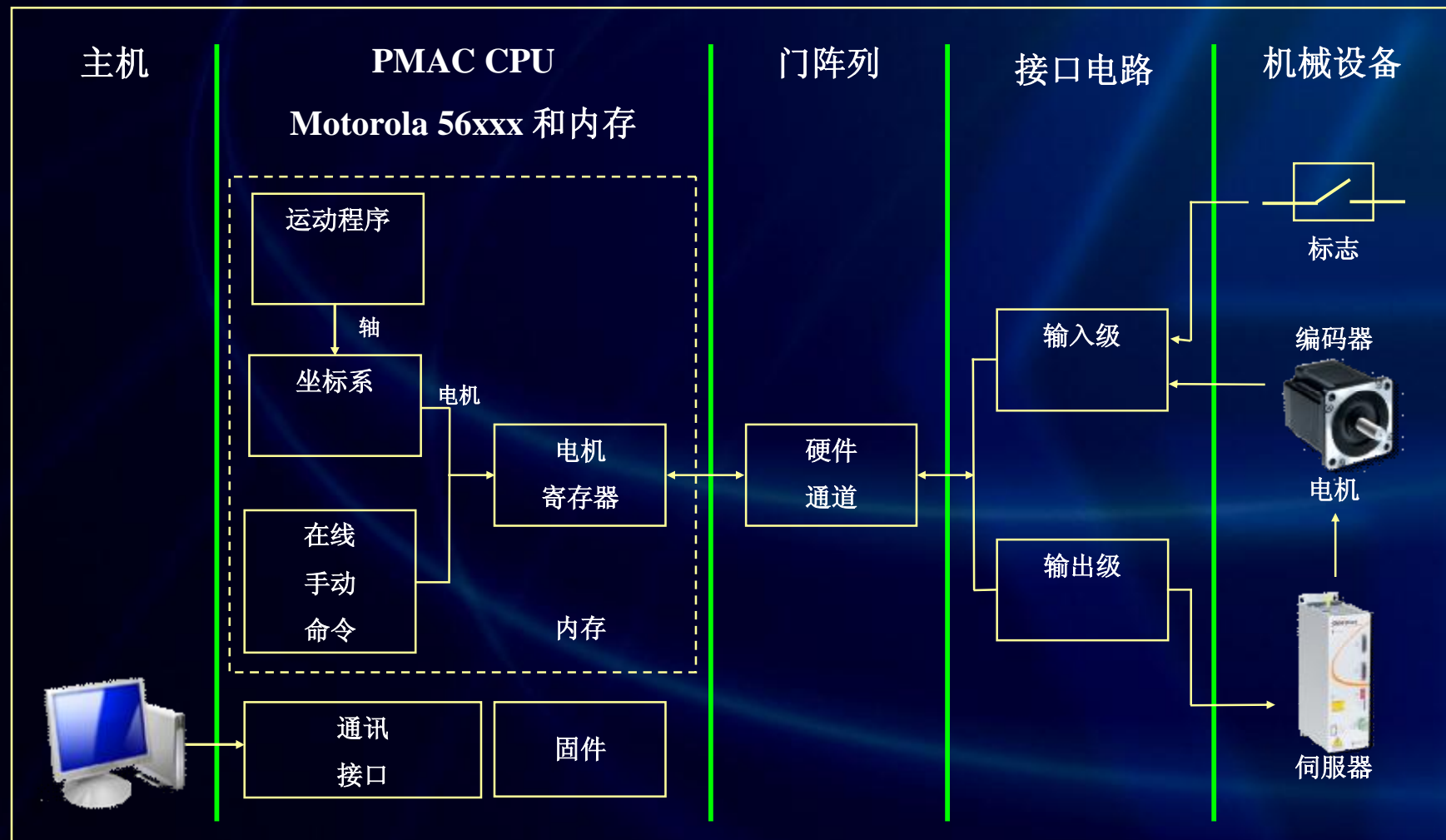


UMAC



Clipper

PMAC系统结构图



定义

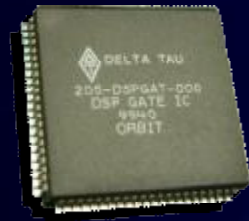
n 硬件通道

- n 正交编码器输入、命令输出和标志的组合
- n PMAC型只能输出DAC $\pm 10V$ 模拟命令信号
- n PMAC2型具有数字、模拟和步进命令输出

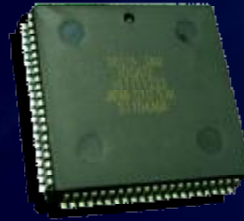
n 门阵列

- n 用户从电机到DSP处理器专用接口芯片
- n 每个芯片具有4轴硬件通道
- n PMAC, PMAC2和MACRO具有不同的门阵列芯片

不同的PMAC卡，不同的门电路



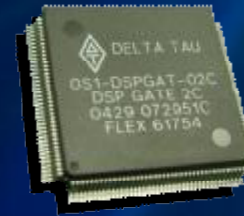
DSPGATE 0



IO GATE



DSPGATE 1



DSPGATE 2



DSPGATE 3

PMAC



PMAC2



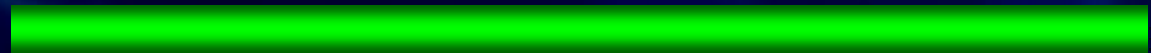
Turbo PMAC



Turbo PMAC2

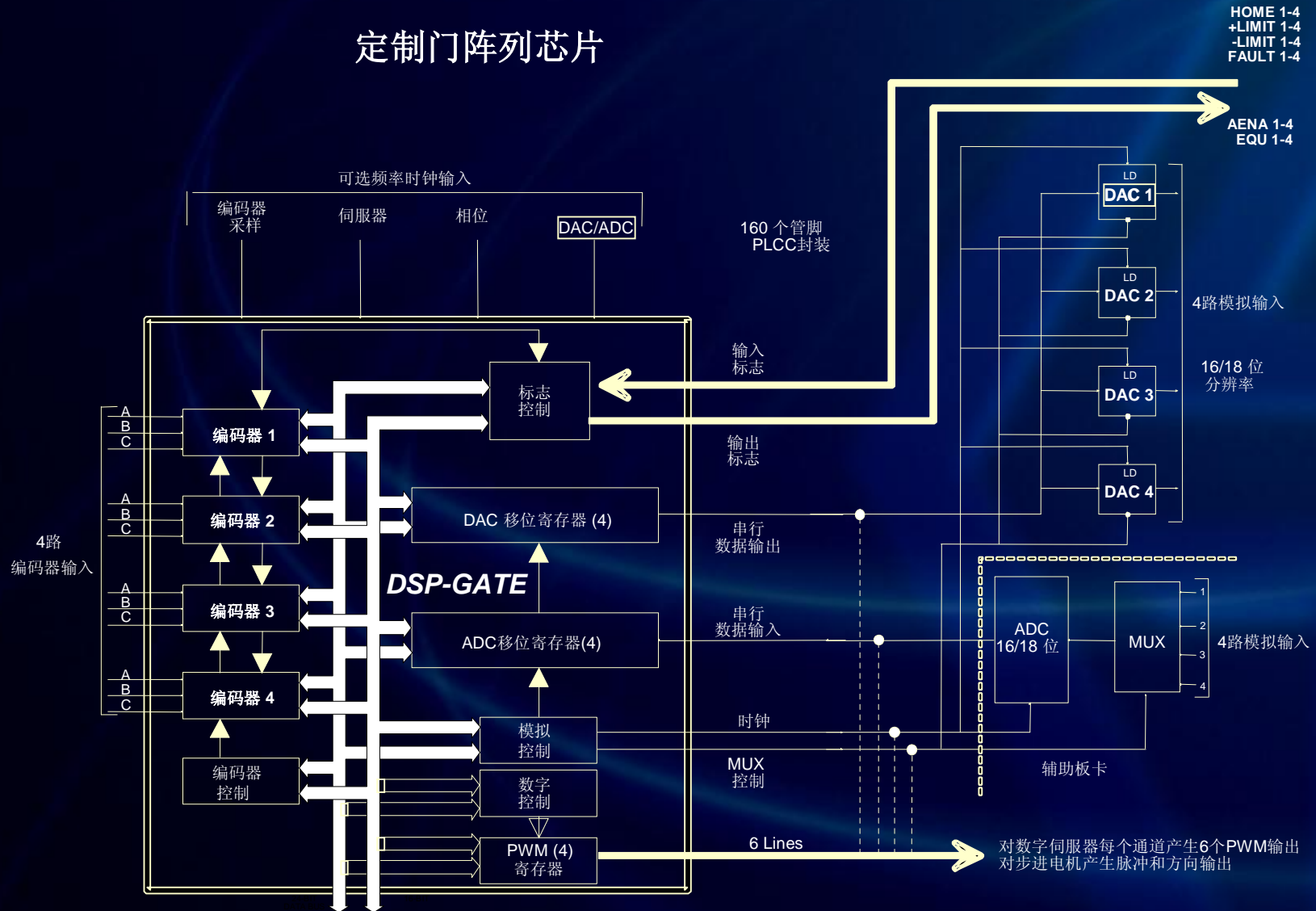


Power PMAC



门阵列结构图

定制门阵列芯片



PMAC概况

Turbo PMAC与Non-Turbo PMAC对比

	Non-Turbo	Turbo
电机数目	8	32
坐标系数目	8	16
前向和逆向的运动学支持	N/A	标准
多步前瞻算法	专用固件	标准
用户变量	4096	32768
多个端口的同步通讯	N/A	标准

PMAC 的多任务性

PMAC 的多任务性

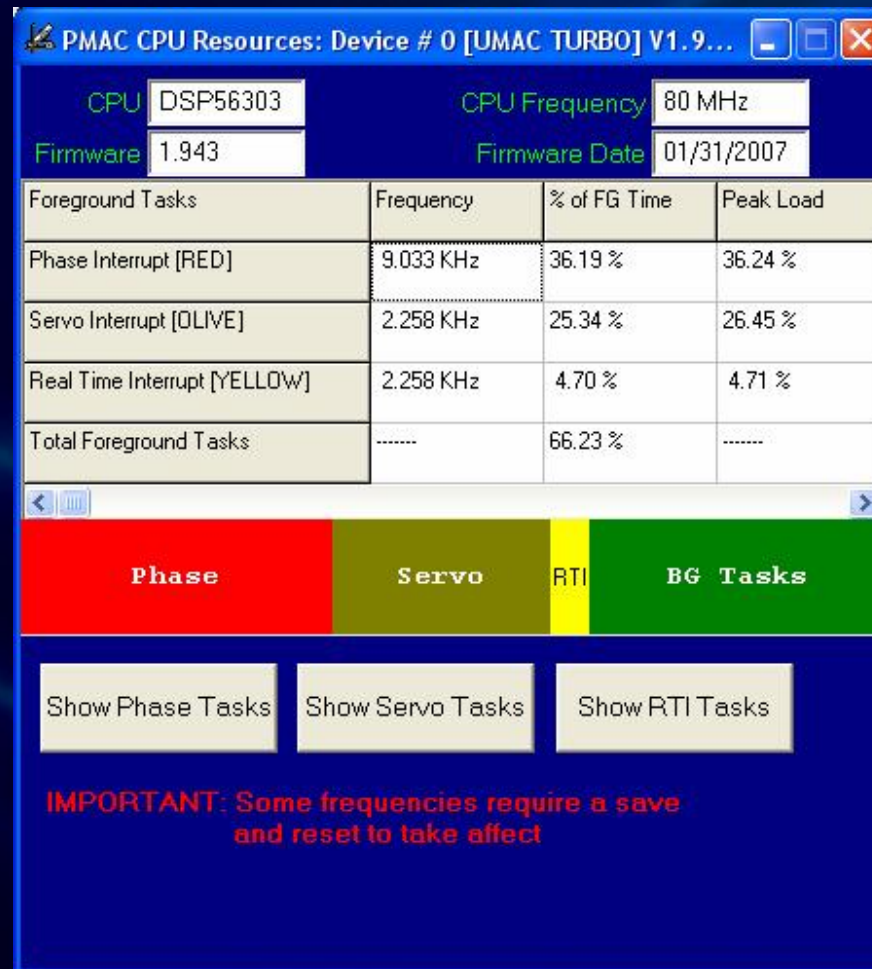
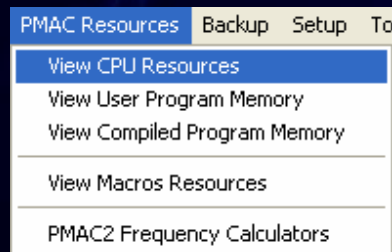
作为多任务、实时的计算机，PMAC具有精密的优先处理方案，以保证重要任务的优先完成，并且保证所有的任务都能够合理、快速地执行。

需要知道：

- n PMAC处理什么任务
- n PMAC怎样完成优先任务

观察CPU的负荷和资源分配

- n 在PEWIN32PRO2软件中，从PMAC Resources菜单下选择View CPU Resources



PMAC可执行什么程序

n 执行运动程序

- n PMAC一次执行一个运动，并运行完到这一步运动相关的所有计算。
- n PMAC总是提前将即将要完成的步骤混合

n 执行PLC程序

- n 在处理器处理时间允许的情况下快速连续地扫描PLC's
- n PLC's对于非同步运行的任务非常有用

PMAC的任务优先级

高

n 单字符 I/O

- n 从串行端口或主机端口输入单字符，或输出单字符

n 相位周期

- n 电流环控制，并对需要换相的电机进行换相计算

n 伺服周期

- n 对每个电机进行伺服控制

n 实时中断

- n 进行运动程序的运动规划
- n 扫描前台PLC(如果使用并激活PLC0和PLCC0),

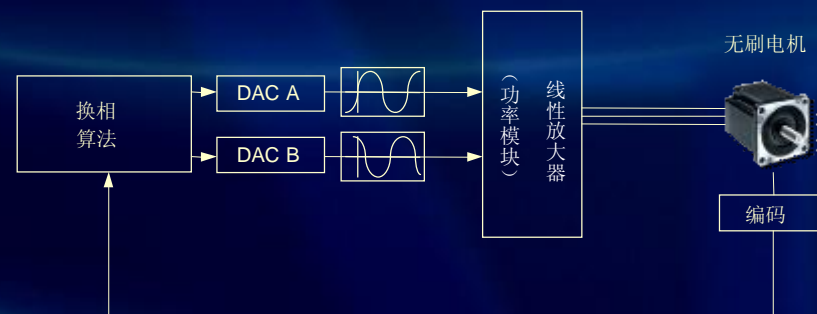
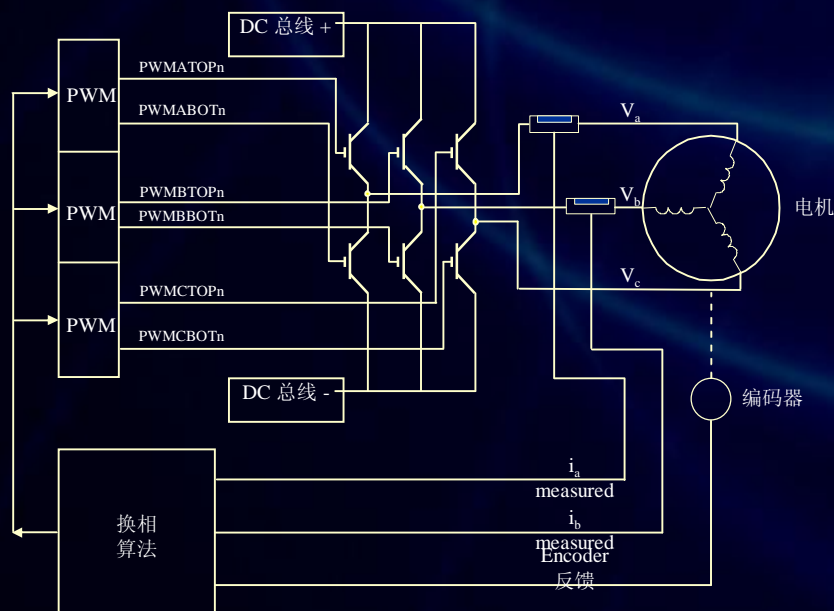
n 后台周期

- n 执行每个激活的PLC和PLCC程序
- n 通讯响应和安全检查

低

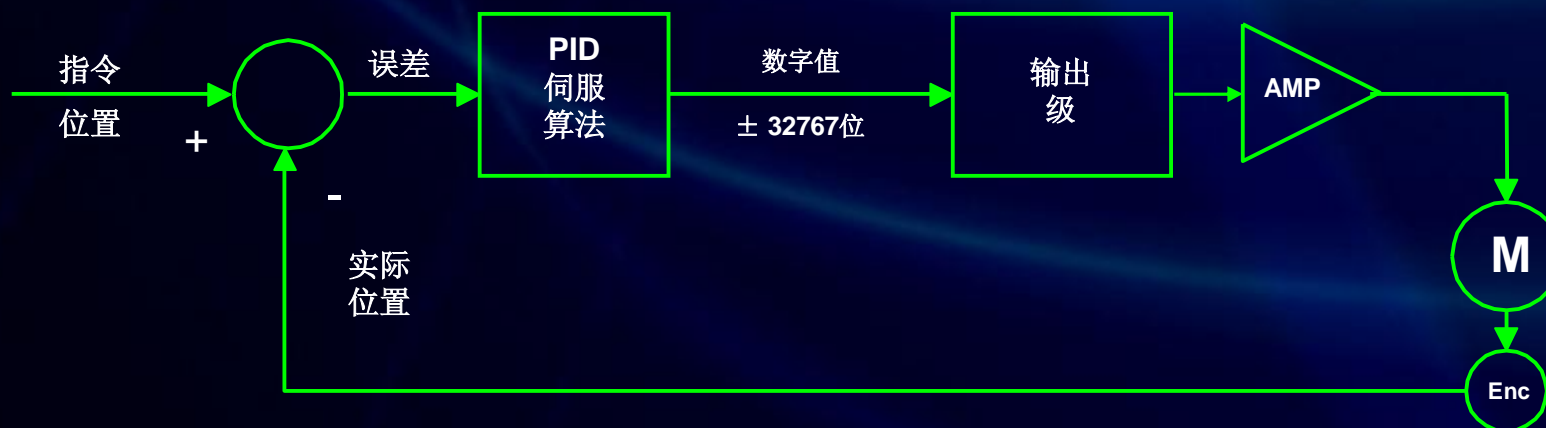
PMAC任务：相位周期

- n PMAC2 / Turbo PMAC2能进行数字或模拟换相
- n 所有PMAC卡的默认速率为9KHz (110usec)
 - n PMAC1由跳线设定速率， PMAC2由软件设定速率
 - n P1的E29-E33， P2的I901， 和TP2的I7m01
- n PMAC1 / Turbo PMAC1只能进行模拟(正弦)换相.
 - n 注意：需要2路DAC输出



PMAC任务：伺服周期

- n 每个电机闭环伺服控制：
 - n 根据设定曲线给出新的给定位置
 - n 从位置反馈装置得到实际位置
 - n 根据两者的差值给出指令输出
- n 指令输出转化为DAC $\pm 10V$ 输出，数字PWM或者脉冲&方向信号
- n 所有PMAC的默认频率为2.25kHz (442usec)
 - n PMAC1由跳线设定速率， PMAC2由软件设定速率
 - n P1的跳线E3-E6， P2的I902， 和TP2的I7m02



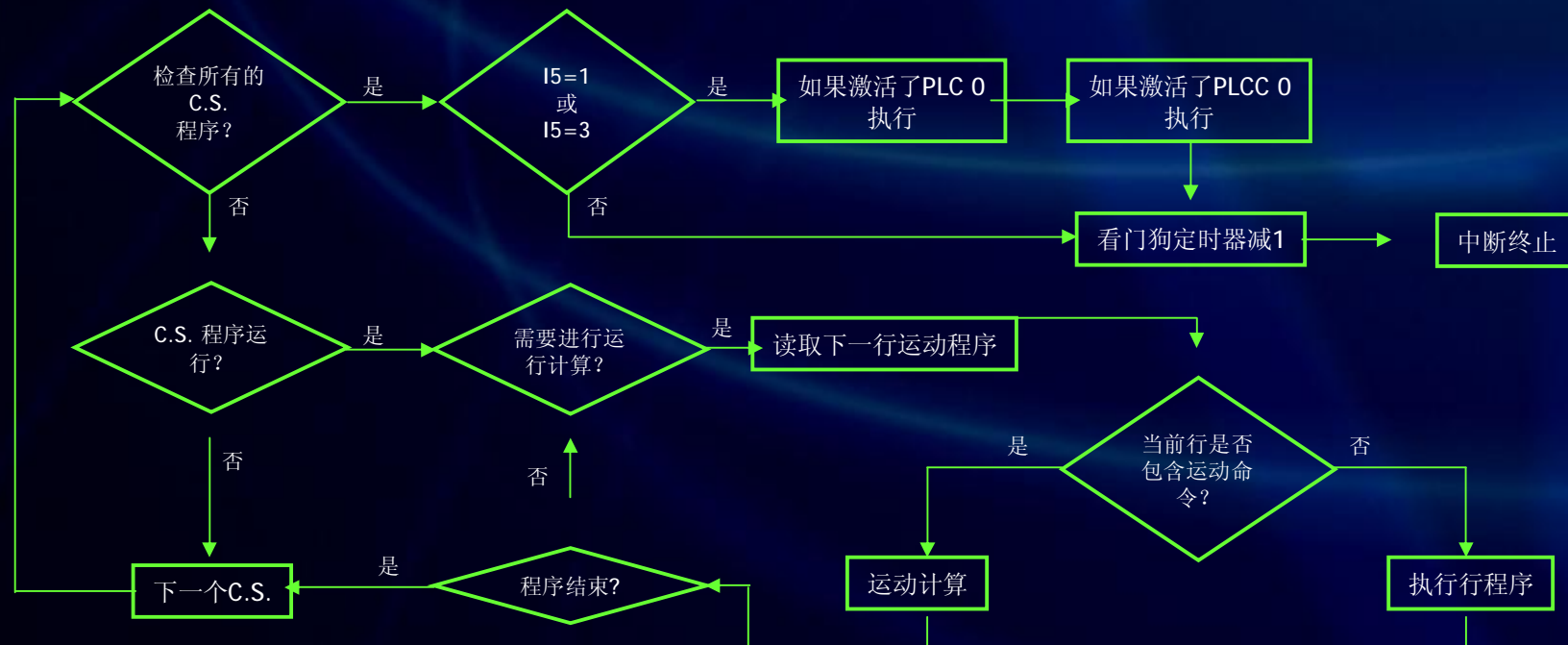
PMAC 任务：实时中断

- n 实时中断 (RTI) 任务的更新率由参数I8控制 (每次在I8+1的伺服周期更新)。

- n 伺服更新任务后立即执行中断

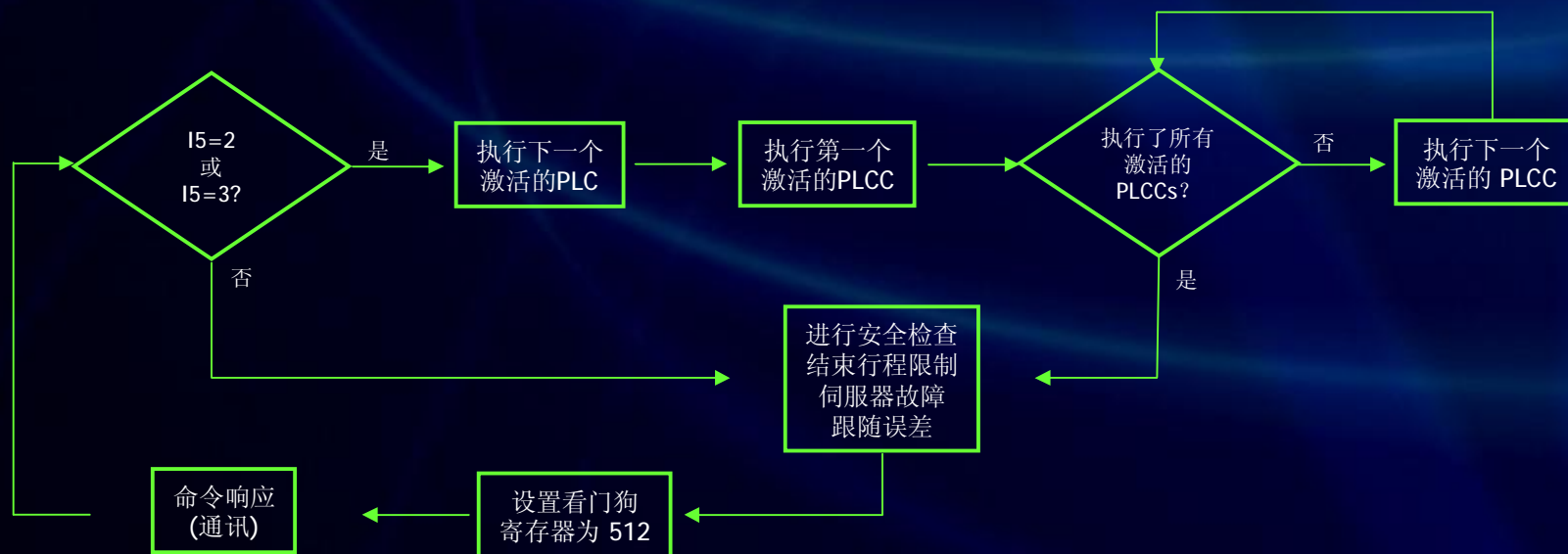
- n 两个主要任务：

- n 启用PLC 0 / PLCC 0执行程序
- n 运动程序



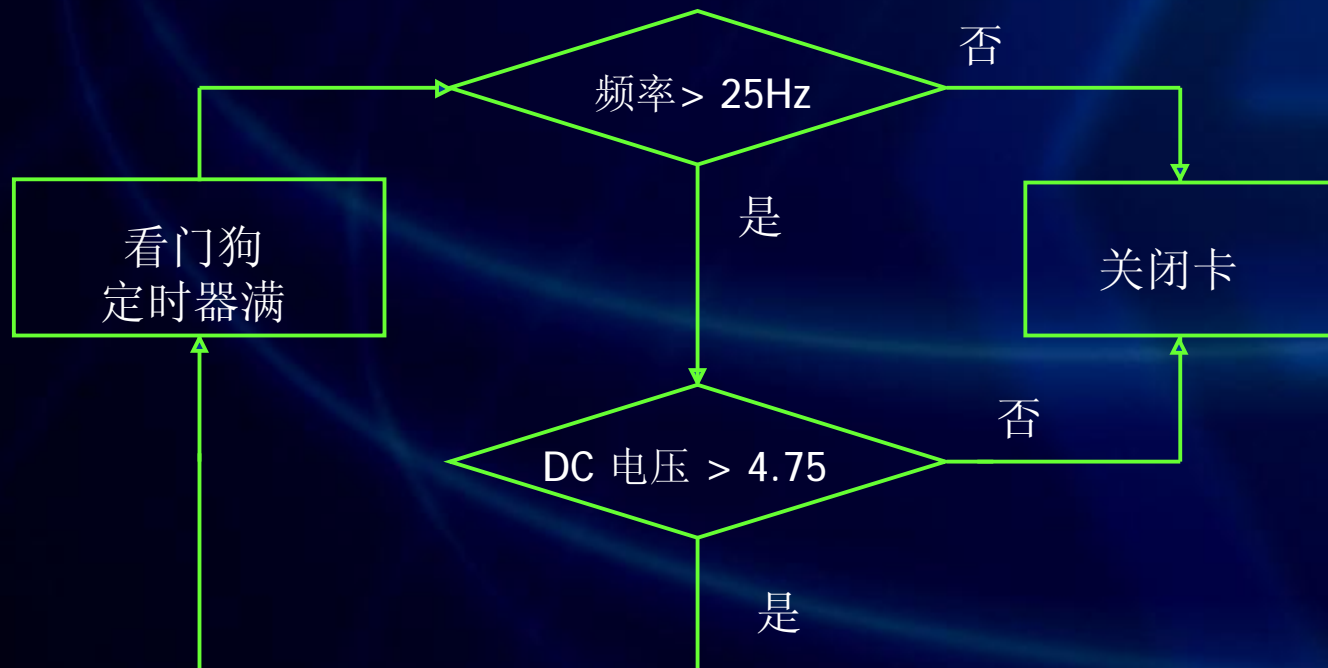
PMAC任务：后台任务

- n (在剩余时间，从高优先级任务开始执行)
 - n 跟随误差限制
 - n 硬件 / 软件超限控制
 - n 伺服器故障
 - n 更新看门狗定时器
 - n PLC 1-31 (一次执行一个程序)
 - n PLCC 1-31 (一次扫描所有程序)
- n PMAC完成这些任务是为了保证其正确地更新。如果在最小频率时没有检查到这一功能，PMAC卡的看门狗将触发。



PMAC 看门狗定时器

PMAC 算法以确保看门狗以25Hz频率被重置，并检查内存芯片有最小4.75V 电压。

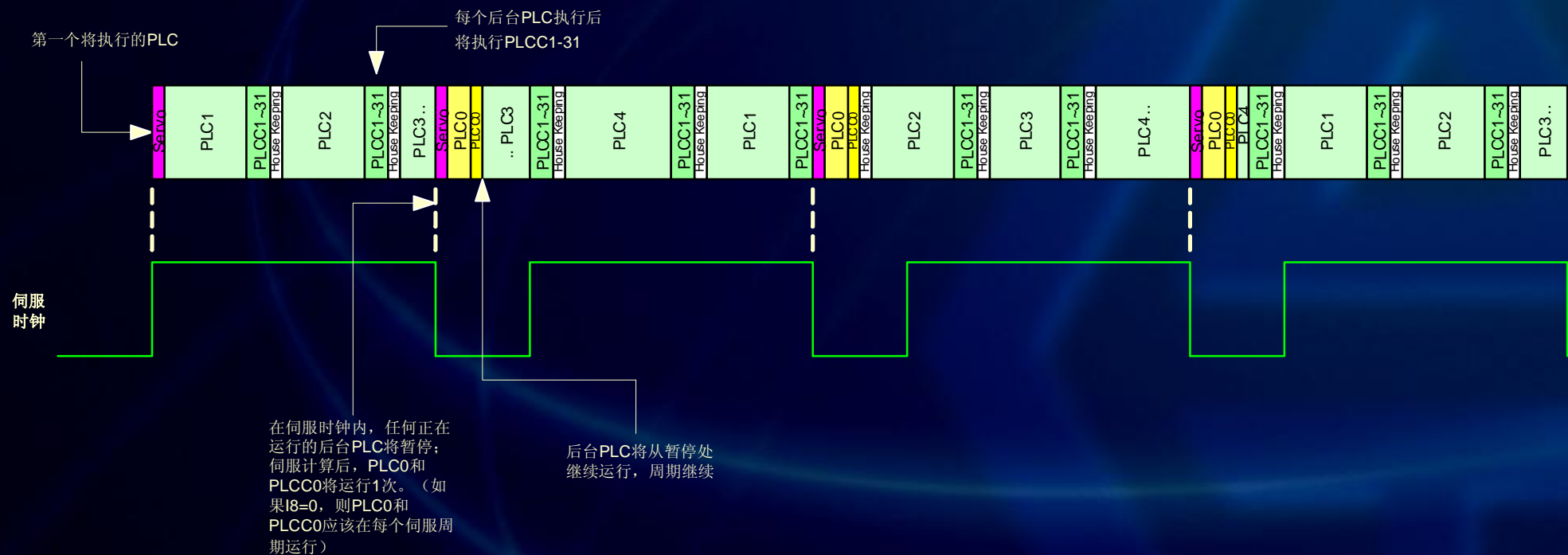


PMAC看门狗定时器

- n 要求电压 $> 4.75\text{ V}$ 并且 $< 5.25\text{ V}$
- n 要求输入频率 $> 25\text{ Hz}$
 - n 开关触发计数器位提供给定时器
 - n 当计数器 > 0 , RTI计数器减1
 - n 后台清理程序重新设置计数器到最大值 (=512)
- n 低电压或者低频率触发定时器
 - n PMAC的红色LED指示灯亮
 - n 命令输出被强制设为0
 - n 设置伺服器使能输出为无效状态
 - n “FEFCO” 输出关闭 (PMAC1)
 - n “WDO” 输出触发 (PMAC2)
- n 重启PMAC电源能清除看门狗故障

PMAC 多任务实例

本例中，PMAC运行程序PLC1—4，PLCC1—31，PLC0 和 PLCC0。



PMAC变量

PMAC的变量和功能

1. I-变量 (Non-Turbo 1024个, Turbo 8192个)

- n 用于初始化和设置
- n 变量功能被预定义
- n 一些卡专用变量
- n 一些电机专用变量
- n 一些坐标系专用变量
- n 一些编码器专用变量

2. P-变量 (Non-Turbo 1024, Turbo 8192)

- n 通用的用户变量
- n 48位浮点形式
- n 全局访问 (不管坐标系)

PMAC变量和它们的功能

3. Q-变量 (Non-Turbo 1024, Turbo 8192)

- n 通用的用户变量
- n 48位浮点形式
- n 坐标系专用变量

4. M-变量 (Non-Turbo 1024, Turbo 8192)

- n 提供内存和I/O访问
- n 用户可定义地址、偏移量和位宽从而进行不同的映射

十六进制

PMAC写十六进制值 (地址或数值)时，用户必须在数字前加"\$"符。

例如：

- n M1000=\$F0 ； 等于十进制的240
- n I125=\$C000 ； non turbo 地址 (十进制的49152)
- n I125=\$78200 ； turbo 地址 (十进制的492032)

十进制	十六进制	二进制
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

例如：

	Nibble				2nd				1st			
位数	7	6	5	4	3	2	1	0	7	6	5	4
二进制	0	1	0	1	1	1	0	0	0	1	1	0
十六进制	5				C							

19 控制十六进制/十进制的显示选择

I-变量类型

n 开关类型

n 数值类型

n 整数型

n 浮点类型

n 地址类型

I-变量分组

I 变量分组	PMAC	PMAC2	Turbo PMAC
卡通用设置	I0-I99	I0-I99	I0-I99
电机变量	I x 00 – I x 86	I x 00 – I x 86	I xx 00 - I xx 99
C.S. 变量	I x 87 – I x 99	I x 87 – I x 99	I sx 00 – I sx 99
伺服芯片设置	I900 – I979	I9 n 0 – I9 n 9	I7 mn 0 – I7 mn 9
伺服芯片时钟	硬件	I900 – I909	I7 m 00 – I7 m 09*
MACRO 芯片设置	-	I990 – I999	I6800 – I6999
编码器转换表	Y:\$720 – Y:\$73F	Y:\$720 – Y:\$73F	I8000 – I8192
备注	x: 电机号或C.S.号	x : 电机号或C.S.号 n : 硬件通道号	* Turbo PMAC1 使用跳线 xx : 电机号 (1-32) sx : C.S.号 + 50 m : 门号 n : 硬件通道号

P-变量

n P-变量是PMAC程序中用于计算的全局和通用的变量。

n 48-位浮点形式

n 从P0到P8191共8192个P-变量

n 用于:

n 1.计算

n `P100=P101*(sin(45))`

n 2. 软件触发

n `IF(M1!= 1 AND P10 = 0)`

P-变量 (续)

如果你想让电机沿位置曲线 $\text{SIN}(\beta) + \text{COS}(\beta)$ 移动，可以按下列方法之一进行操作：

采用程序中预先计算好的点轨迹

```
X1
X1.0173
X1.0343
.
.
X0.9824
X1
```

采用方程式动态产生点轨迹

```
P1=0
WHILE (P1<361)
    P2=SIN(P1) +COS(P1)
    X(P2)
    P1=P1+1
ENDWHILE
```

程序运算能力

- n PMAC的DSP的计算能力支持运动程序中大量的运算

```
OPEN PROG 1 CLEAR
WHILE(1=1)
  IF(P1>0)
    P2=SIN(P1)+COS(P1)
    p3=2
    IF(P1>3)
      P2=SIN(P1)+COS(P1)
      P3=2
    W
    W
      IF(P1>99)
        P2=SIN(P1)+COS(P1)
        p3=99
      ENDIF
    W
    W
      ENDIF
    ENDIF
    X2000
    P1=P1+1
  ENDWHILE
CLOSE
```

Q-变量

- n Q-变量是PMAC程序中用于用户计算的通用变量
- n Q-变量是坐标系专用变量
- n 采用Q-变量使多C.S.的变量管理简化

Q-变量内存映射图

n 物理内存的访问通过键入Q(序号)，根据当前的定址坐标系改变内存。

n &1 Q0 访问位置 \$6000

n &2 Q0 访问位置 \$6200

n &15 Q0 访问位置 \$7C00

n &16 Q0 访问位置 \$7E00

n 这种定址法简化了多坐标系应用时的内存管理。如果在16个坐标系下执行16个运动程序，所有的程序可以使用相同的变量序号Q0到Q127，减少了繁琐，并且没有内存冲突。

Q-变量 内存 Mapping

n 物理内存能被不同的坐标系的不同Q变量访问（在不同坐标系，访问同一内存地址的Q变量不相同）

n	&1 Q0	访问位置	\$6000
---	-------	------	--------

n	&2 Q4096	访问位置	\$6000
---	----------	------	--------

n	&15 Q2560	访问位置	\$6000
---	-----------	------	--------

n	&16 Q512	访问位置	\$6000
---	----------	------	--------

n 随着坐标系数目的增加，每个坐标系非重叠 Q-变量的数目减少

Q-变量 映射

访问地址	1	2	4	8	16
\$6000	C.S.1	C.S.1	C.S.1	C.S.1	C.S.1
\$61FF					C.S.9
\$6200				C.S.5	C.S.5
\$63FF					C.S.13
\$6400			C.S.3	C.S.3	C.S.3
\$65FF					C.S.11
\$6600				C.S.7	C.S.7
\$67FF					C.S.14
\$6800		C.S.2	C.S.2	C.S.2	C.S.2
\$69FF					C.S.10
\$6A00				C.S.6	C.S.6
\$6BFF					C.S.15
\$6C00			C.S.4	C.S.4	C.S.4
\$6DFF					C.S.12
\$6E00				C.S.8	C.S.8
\$6FFF					C.S.16
\$7000			C.S.4	C.S.4	C.S.4
\$71FF					C.S.12
\$7200				C.S.8	C.S.8
\$73FF					C.S.16
\$7400				C.S.8	C.S.8
\$75FF					C.S.16
\$7600					C.S.8
\$77FF					C.S.16
\$7800				C.S.8	C.S.8
\$79FF					C.S.16
\$7A00					C.S.8
\$7BFF					C.S.16
\$7C00				C.S.8	C.S.8
\$7DFF					C.S.16
\$7E00					C.S.8
\$7FFF					C.S.16

M-变量

- n M-变量用于访问PMAC内存和I/O点
- n M-变量没有预定义。用户必须将它们指向有效的PMAC 地址。
例如：`M9->Y:$FFC2,8,8 ;JOPTO`端口输出字
- n M-变量一旦定义好后，可用于设置状态、计算和判别触发

M-变量的定义类型

M-变量定义的地址前缀，可以是下列类型：

M9->Y:\$FFC2,8,8

X: X内存中的1-24位的固定地址位

Y: Y内存中的1-24位的固定地址位

D: 同时占用X和Y内存的48位固定地址位

L: 同时占用X和Y内存的48位浮点地址位

DP: 32位的固定地址位(X和Y内存的低16位)(双端口RAM使用)

F: 32位的浮点地址位(X和Y内存的低16位)(双端口RAM使用)

简单的M变量定义

n X: {地址}, {偏移量}, {宽度}, {格式}

n Y: {地址}, {偏移量}, {宽度}, {格式}

偏移量

一起始的位号

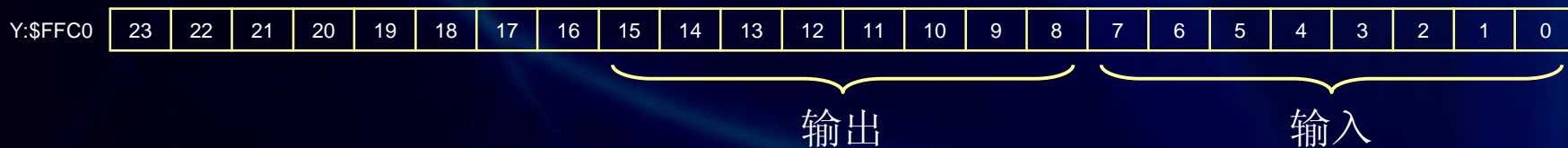
宽度

—缺省宽度为1；可以是 1, 4, 8, 12, 16, 20, or 24 位宽

格式

— 缺省格式为U (无符号的); 可以为有符号格式S

PMAC1 内存位置Y :\$FFC2 对应于JOPTO端: 8路输入和8路输出



```
M9->Y:$FFC2,8,8      ;JOPTO端输出字
M19->y:$FFC2,0,8      ;JOPTO端输入字
```

推荐的M-变量定义示例

推荐的M-变量在PMAC的软件参考手册中(可在PEWIN Pro2 Suite中, 选取菜单Configure中的M-Variables选项下载推荐配置)

M0->x:\$0000,0,24	;指向伺服时钟
M1->Y:\$FFC2,8,1	;指向卡的输出 1
M9->Y:\$FFC2,8,8	;指向卡的输出 1-8
M102->Y:\$C003,8,16,s	;指向DAC 1输出
M197->X:\$0806,0,24,s	;指向进给修调
M120->Y:\$C000,20,1	;指向Home标志轴1
M172->L:\$082B	;指向#1 变量 手动
M1000->*	;自参照 (指向自身的内存位置)

M-变量的使用

M1=1 ;开通卡的输出1
M9=45 ;开通卡的输出1,3,4,6; 关闭卡的输出2,5,7,8
 ;45=00101101 [二进制]=\$2D [十六进制]

数组

n 利用一系列连续的变量建立数组，如P1到P20

n 读数组，在Turbo和非Turbo PMAC中有效：

P1 = 10	; 本例中P1为数组的索引变量
P3 = P(P1)	; 相当于P3 = P10
X(P(P1))	; 根据索引变量为P1的数组移动轴X

n 写数组，在Turbo和非Turbo PMAC中有效：

Array Pointer →	M34->L:\$1001	; non-Turbo PMAC卡中P1的地址位置
Array Index →	M35->Y:\$BC22,0,16	; non-Turbo PMAC卡中M34的定义字
	M34 = 5	; 相当于P1 = 5
	M35 = M35 + 1	; 索引加1，则M34指向P2
	M34 = 6	; 相当于P2 = 6

n 写数组，仅在Turbo PMAC中有效：

P1 = 15	; 本例中P1为数组的索引变量
P(P1) = 5	; 相当于P15 = 5

M-变量的小测验

- n 怎样定义一个M-变量以读取卡的输入2？数值类型
- n 怎样定义一个M-变量以读取卡的输入1至8？
- n 如何开通输出1、2和8？
- n 如何获知输入1、2和8的开通状态？
- n 何时以及何处定义M-变量？

M-变量的实例操作

- n 在演示软件中为每个数字输入和输出分别定义M-变量
- n 在观察窗口监测输入的M-变量
- n 定义M-变量，命令所有的输出作为一个字节输出
- n 键入**SAVE**命令，将M-变量的定义存入FLASH
- n 键入**\$\$\$**，确保M-变量的定义都保存好

PMAC的设置参数

设置 / 初始化I-变量

Non-Turbo PMAC I-变量组

n I0 - I99	常规的卡设置
n I100 - I186	电机1的设置
n I187 - I199	坐标系1的设置
n I200 - I286	电机2的设置
n I287 - I299	坐标系2的设置
n
n I800 - I886	电机8的设置
n I887 - I899	坐标系8的设置
n I900 - I999	硬件设置
n I1000 - I1023	MACRO和杂项的设置

Turbo PMAC I-变量组

n	I0 - I99	常规的卡设置
n	I100 - I199	电机1的设置
	...	
n	I3200 - I3299	电机32的设置
n	I3300 - I3349	电机1的补充设置
	...	
n	I4850 - I4899	电机32的补充设置
n	I4900 - I4999	系统配置的状态
n	I5000 - I5099	数据采集 & ADC的多路分配控制
n	I5100 - I5199	坐标系1的设置
	...	
n	I6600 - I6699	坐标系16的设置
n	I6800 - I6849	MACRO 芯片 0的硬件设置
	...	
n	I6950 - I6999	MACRO 芯片 3的硬件设置
n	I7000 - I7099	伺服 芯片 0的硬件设置
	...	
n	I7900 - I7999	伺服 芯片 9的硬件设置
n	I8000 - I8191	编码转换表的设置

卡硬件设置 I-变量

- n** I7m00: 最大相位和PWM频率控制
最大相位频率(kHz)= $117,968/(2*I7m00+3)$
PWM频率(kHz)= $117,968/(4*I7m00+6)$
注: m是从0开始的整数, 对应门阵列的编号
- n** I7m01: 相位时钟频率控制
相位频率(kHz)=最大相位/(I7m01 + 1)
- n** I7m02: 伺服时钟频率控制
伺服频率(kHz)=相位/(I7m02 + 1)
- n** I7m03: 硬件时钟频率控制
串行编码器时钟(Encoder SCLK), DA转换时钟(DAC_CLK), AD采样时钟(ADC_CLK), 脉冲频率调制时钟(PFM_CLK)

卡硬件设置 I-变量 (续)

- n I7m04: PWM死区时间/PFM脉冲宽度控制
死区时间(msec)= $0.135 * I7m04$
脉冲宽度 (msec)=PFM_CLK周期 * I7m04
- n I7m05: DAC选通控制字 (ACC-8E的\$7FFFC0)
- n I7m06: ADC选通控制字 (Geo Amps的\$3FFFFFF)
- n I7m07: 相位/伺服时钟方向
0: 内部相位时钟; 内部伺服时钟 (源)
3: 外部相位时钟; 外部伺服时钟 (接受者)

硬件(门阵列)通道专用的 I-变量

n I7mn0:

n号编码器的解码控制

0: 脉冲 & 方向 CW

4: 脉冲 & 方向 CCW

1: x1 正交CW

5: x1 正交CCW

2: x2 正交CW

6: x2 正交CCW

3: x4 正交CW

7: x4 正交CCW

8: 内部脉冲 & 方向

12: MLDT 脉冲计时

11: x6 Hall CW

15: x6 Hall CCW

注: n是从1到4的整数, 对应m号门阵列的通道号

n I7mn1:

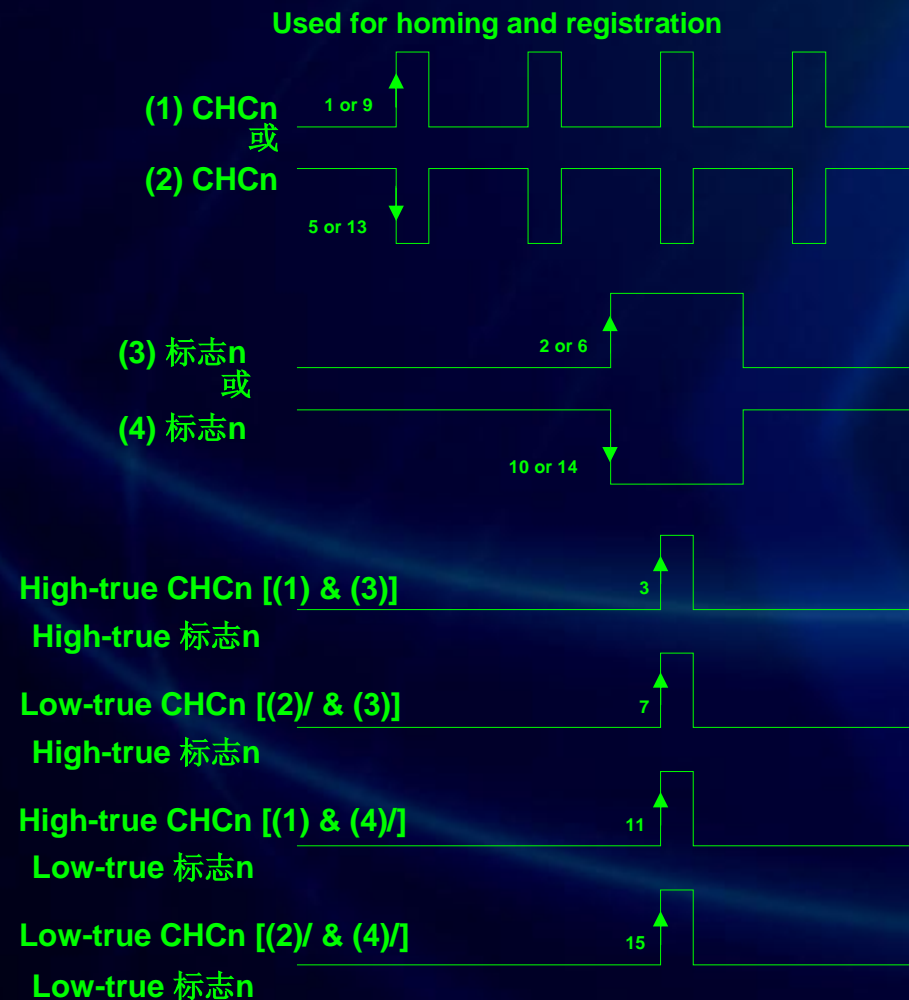
n号通道位置比较源的选择

0: 使用n号编码器

1: 在IC上使用1号编码器

硬件(门阵列)通道专用的 I-变量(续)

n I7mn2: n号编码器的捕获控制



硬件(门阵列)通道专用的 I-变量(续)

n I7mn3: n号编码器捕获标志信号的选择

- 0: HMFLn (回零)
- 1: PLIMn (正限位)
- 2: MLIMn (负限位)
- 3: USERn (用户信号)

n I7mn4: n号编码器索引输入的选择

- 0: 编码器索引直接进入位置捕获电路
- 1: 编码器索引和正交信号逻辑组合后进入位置捕获电路 (可在一个正交状态宽度获得精确的索引信号)

n I7mn5: n号正交编码器索引输入状态/解码控制

- 0: “高-高” 正交状态时编码器索引作为位置捕获信号
- 1: “低-低” 正交状态时编码器索引作为位置捕获信号

硬件(门阵列)通道专用的 I-变量(续)

n I7mn6: n号输出模式的选择(A,B,C三个输出)

0: A&B – PWM	C – PWM
1: A&B – DAC	C – PWM
2: A&B – PWM	C – PFM
3: A&B – DAC	C – PFM

n I7mn7: n号输出的反向控制

0: A&B – 不反向	C -不反向
1: A&B –反向	C -不反向
2: A&B -不反向	C –反向
3: A&B –反向	C -反向

n I7mn8: n号输出的PFM方向控制

0: 不反向的方向输出
1: 反向的方向输出

电机的I-变量组

- n** lxx00 - lxx10 电机 xx 的参数定义
- n** lxx11 - lxx19 电机 xx 的安全变量
- n** lxx20 - lxx29 电机 xx 的运动变量
- n** lxx30 - lxx35 电机 xx 的PID调节参数
- n** lxx36 - lxx39 电机 xx 的陷波/低通滤波器
- n** lxx40 - lxx59 电机 xx 的扩展伺服算法参数
(non-Turbo 需要含Option-6的固件)

- n** lxx60 - lxx69 电机 xx 的标准伺服环调节参数
- n** lxx70 - lxx84 电机 xx 的换相和寻相参数
(仅在lxx01=1时才有意义)

- n** lxx85 - lxx86 电机 xx 的齿隙参数

注: xx是从1到32的整数(Non-Turbo 1~8), 对应xx号电机

基本的电机定义 I-变量

n Ixx00

电机 xx 使能变量

n Ixx00=0

电机 xx 禁用

PMAC不执行该电机的伺服计算

PMAC不进行电机位置报告

n Ixx00=1

电机 xx 激活

PMAC进行该电机的伺服计算

不一定需要启用

基本的电机定义 I-变量(续)

n Ixx01

电机 xx 换相

n Ixx01=0

不用PMAC进行换相(换相功能在伺服器上)

PMAC不进行无寻相计算

A,B,C中只有A有输出

n Ixx01=1

PMAC进行换相

每次相位更新周期中进行定相计算

A,B,C中有2路模拟或者3路PWM输出

必须使用Ixx70-Ixx84

与PMAC地址相关的I-变量 (Non-Turbo)

- n 低16位(4个十六进制数)表示地址
- n 当高8位为0时，则地址为标准模式
- n 当一个或多个高位=1时，则地址为不同的模式
- n 当I9=2或3时：PMAC按十六进制表示变量值

Hex (\$)	0				1				C				0				0				3			
Binary	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1		

相关变量：

I21-I44

Ixx02, Ixx03, Ixx04, Ixx05, Ixx10, Ixx25,
Ixx81, Ixx82, Ixx83, Ixx93

与电机地址相关的I-变量

PMAC vs. Turbo PMAC

n lx02 命令输出地址和方式

n lx03 位置环反馈地址和方式

n lx05 主电机位置地址和方式

n lx06 主电机跟随允许

n lx10 上电启动绝对位置信息获取地址和方式

n lx25 标志信号地址和方式

n lx81 上电启动相位信息的获取地址和方式

n lxx02 命令输出地址
lxx96 命令输出方式

n lxx03 位置反馈地址
lxx97 位置捕获方式

n lxx05 主电机位置地址

n lxx06 主电机跟踪启动和方式

n lxx10 上电启动绝对位置信息获取的地址
lxx95 上电启动绝对位置信息获取的格式

n lxx25 标志信号地址
lxx24 标志信号方式

n lxx81 上电启动相位信息的获取地址
lxx91 上电启动相位信息的获取格式

电机地址的I-变量

n lxx02

命令输出地址

指定第一个命令寄存器(1个、2个或3个)的地址
通常为DAC, PWM, PFM, 或MACRO的输出寄存器

n lxx03

位置环反馈地址

指定(X) 寄存器为位置环反馈地址
(通常在转换表中)

n lxx97

位 0 指定软件位置触发捕获
(非硬件捕获)

位 1 指定在警告跟随误差上触发
(非捕获触发)

电机地址的I-变量(续)

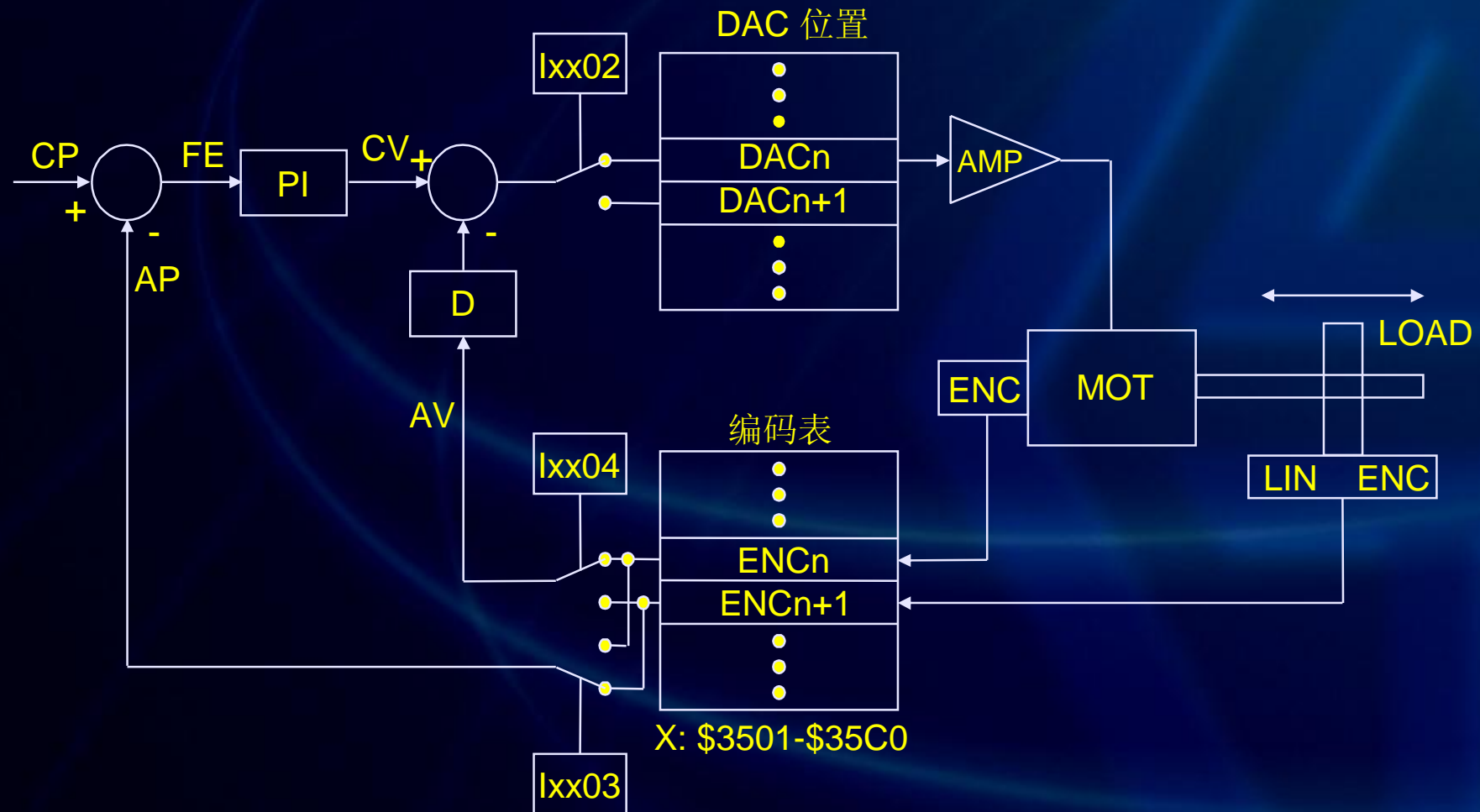
n Ixx04 速度环反馈地址

指定(X) 寄存器为速度环反馈地址
(通常在转换表中，寄存器必需包含位置数据)

n 通常指定与Ixx03相同的寄存器。

n 如果不同，例如双反馈场合，则Ixx03采用负载上的反馈，Ixx04采用电机上反馈。

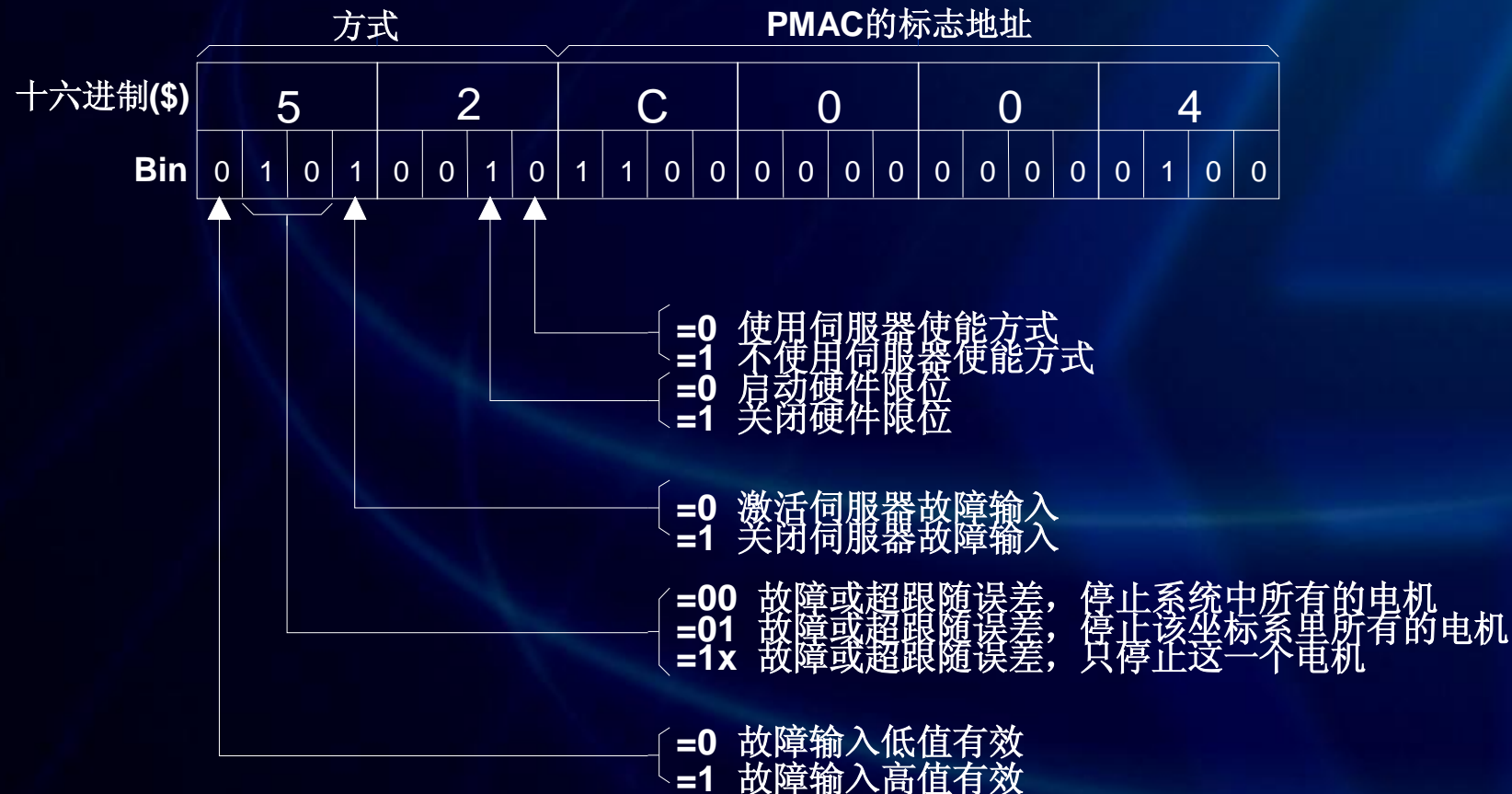
命令/反馈 I-变量的结构图



电机地址的I-变量(续)

n lxx25

电机 xx 的标志位地址和方式 (Non-Turbo)



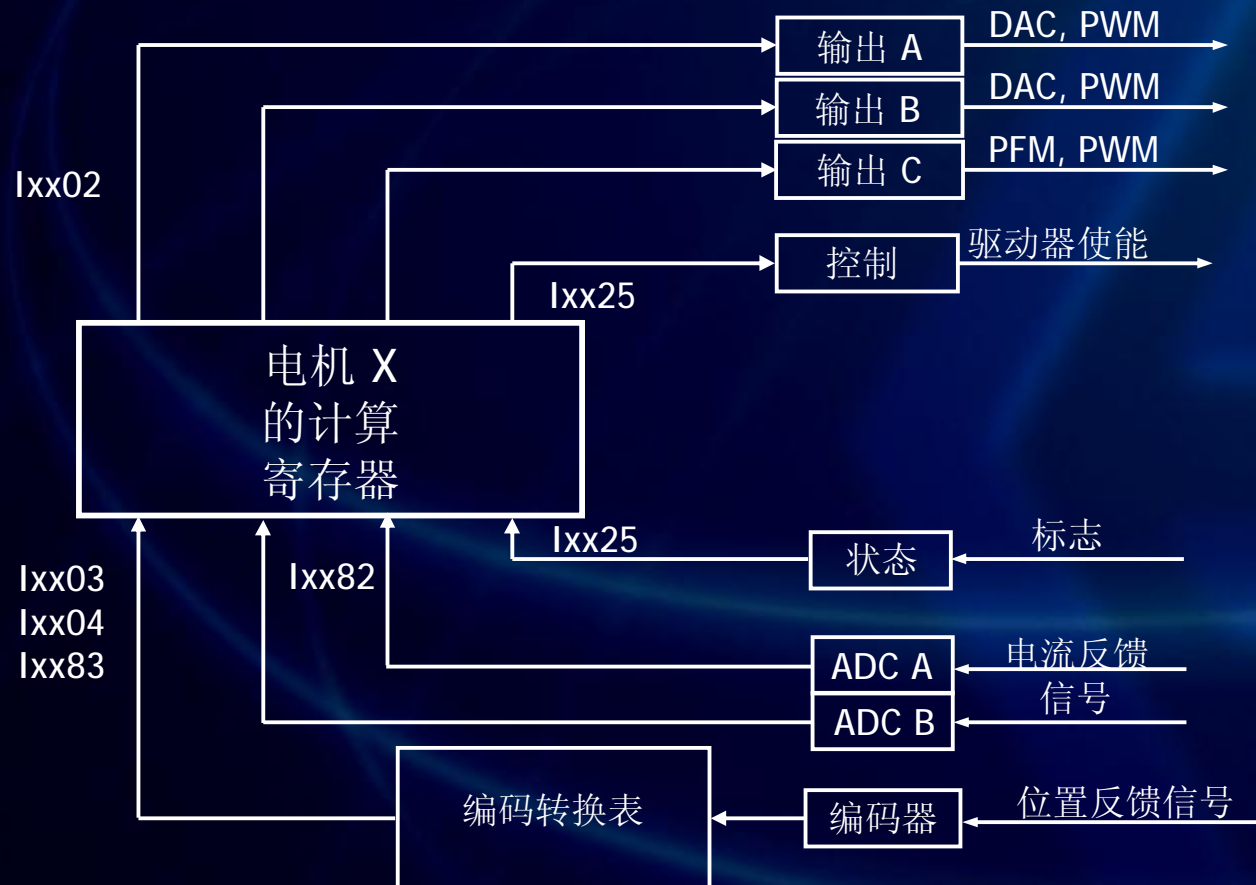
使用回零时, 所用标志位和增量式编码器必需在门阵列芯片的同一通道

电机地址的I-变量(续)

- n lxx25 电机 xx 的标志地址和方式 (Turbo)
- n lxx24 电机 xx 的方式



PMAC直接寄存器映射



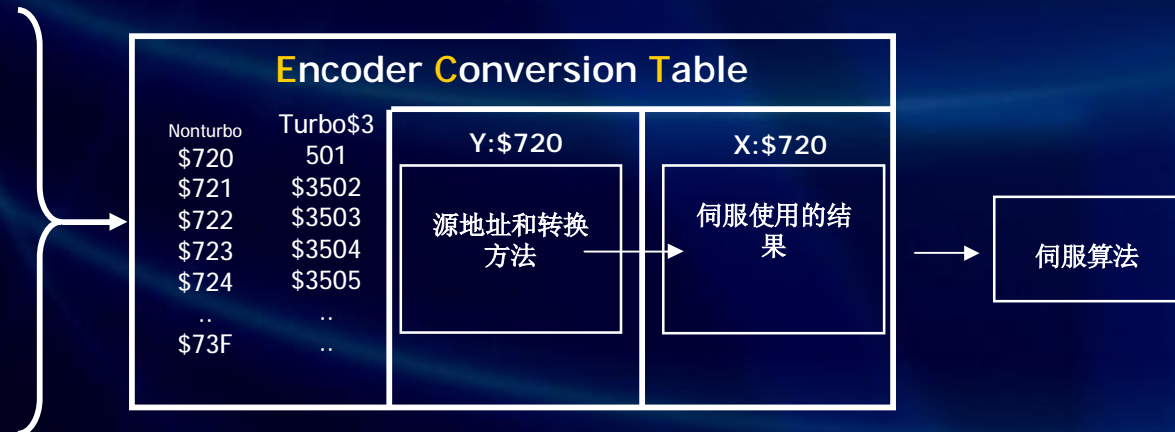
编码转换表

n Turbo PMAC的伺服算法用两步处理反馈信息：

- n 第一步利用硬件寄存器 (编码计数器，定时器和A/D寄存器)，在伺服中断时，无直接软件干扰下连续锁存数据。
- n 第二步采用称为**编码转换表 (ECT)**形式的数据结构，预处理锁存在寄存器中的信息。

n 缺省模式下，ECT转换正交增量式编码器的数据

- 正交, 增量, 编码器
- 并行两进制反馈
- 光栅尺反馈
- 模拟反馈
- 4096倍正弦插补反馈
- SSI 编码器输入
- 安川 或 三菱 绝对值编码器
- 磁感应式位移反馈输入



编码转换表的设置

Turbo PMAC 编码器转换条目

编码转换行	I-变量	地址	Line	I-变量	地址
1	I8000	\$3501	9	I8008	\$3509
2	I8001	\$3502	10	I8009	\$350A
3	I8002	\$3503	11	I8010	\$350B
4	I8003	\$3504	12	I8011	\$350C
5	I8004	\$3505	13	I8012	\$350D
6	I8005	\$3506	14	I8013	\$350E
7	I8006	\$3507	15	I8014	\$350F
8	I8007	\$3508	16	I8015	\$3510

编码转换表中行变量(也是一种I-变量) 结构和意义

位	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
数字	1				2				3				4				5				6			
内容	方法				*	源地址																		

例如:

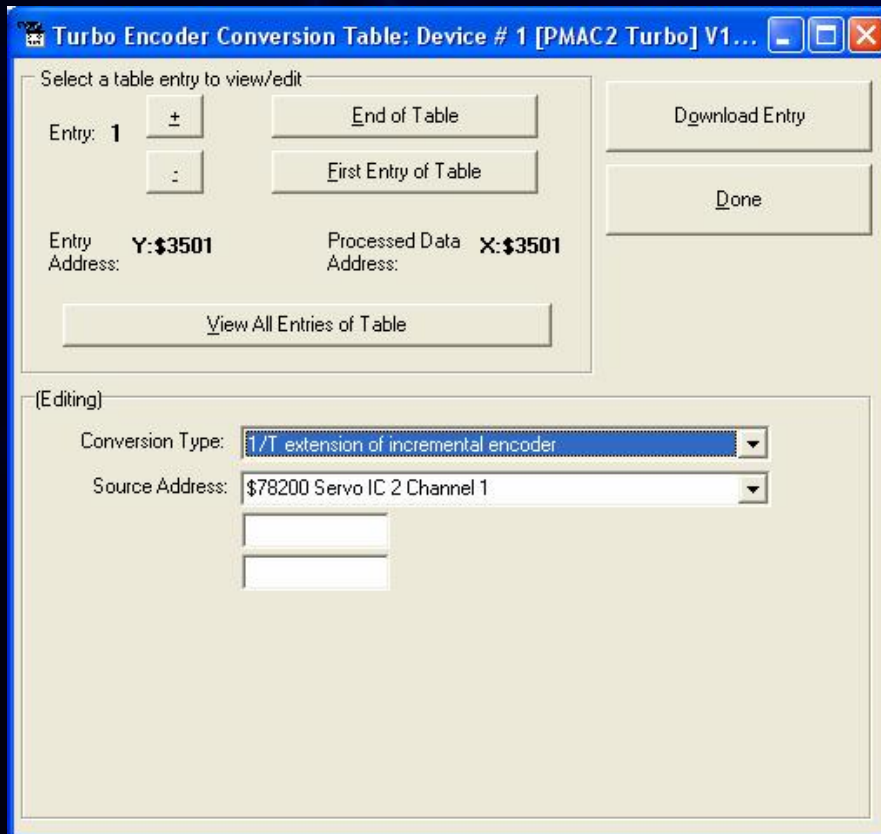
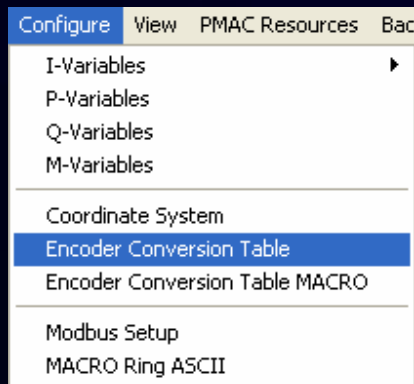
I8000=\$078000 或 wy:\$3501,\$078000

; 编码转换表的第一行被设置为去读取伺服门阵列0号芯片的1通道的原始数据(\$078000)

I8001=\$078004

;编码转换表的第二行(对应I变量的本身的地址Y:\$3502) 设置为读取伺服门阵列0号芯片的2通道的正交原始数据(\$078004)

编码转换表的设置



- n Turbo UMAC缺省设置解释:
 $I8000 = \$078200$; Y:\$3501
 0意味着对0号伺服门阵列1号通道编码器的软件1/T扩展
 78200 意味着0号伺服门阵1通道编码器在UMAC中的地址
 $I8001 = \$078208$; Y:\$3502
 0意味着0号伺服门阵2号通道编码器的软件1/T(UMAC)
 78208 意味着0号伺服门阵2通道编码器在UMAC中的地址
- n 对于增量式编码器，源地址必需是门阵列编码计数器之

注意：上页是针对Turbo PMAC，这里是Turbo UMAC

UMAC 门阵列 编码器地址

ENC1:	\$078200	ENC5:	\$078300
ENC2:	\$078208	ENC6:	\$078308
ENC3:	\$078210	ENC7:	\$078310
ENC4:	\$078218	ENC8:	\$078318

1/T extension of incremental encoder
 Acc-28 A/D register (no rollover)
 Acc-28 A/D register with integration (no rollover)
 Parallel pos from Y word with no filtering
 Parallel pos from Y word with filtering
 Timebase conversion with scaled differentiation
 Triggered timebase conversion
 Parallel pos from Y/X word with no filtering
 Parallel pos from Y/X word with filtering
 Parallel extension of incremental encoder
 No extension of incremental encoder
 Exponential filter
 Summing Of conversion table entries
 High res. interpolator (Accessory 51P) PMAC1 style
 High res. interpolator (ACCs. 51C,E,P2,S) PMAC2 style
 Byte-wide parallel Y word with no filtering
 Byte-wide parallel Y word with filtering
 Tracking filter
 High res. interpolator diagnostic
 Resolver
 End of Table

电机 xx 的安全变量

- n lxx11: 致命跟随误差限制设定
(单位: 1/16 编码器计数, ct)

=0 时为无效

当设定值被超过时:

- n 退出坐标系程序
- n 电机按lxx24的设置停止运行
- n 位22和21
 - n (1x): 仅对这一电机
 - n (01): 同一坐标系下所有电机
 - n (00): 所有系统中电机

电机 xx 的安全变量 (续)

n lx12: 警告跟随误差限制设定
(单位: 1/16编码器计数,ct)

=0 时为无效

n 当跟随误差超过设定时:

- n** 设置电机和C.S.的状态位
- n** 可以用于设置输出和中断
- n** 可以用做触发条件

电机 xx 的安全变量 (续)

- n lxx13: 软件的正限位设定
- n lxx14: 软件的负限位设定
(单位: 编码器计数,cts)

=0 时为无效

- n 当超出时:
 - n 退出运动程序或电机的运动
 - n 电机按lxx15设置的减速度减速

- n lxx15: 越限或退出时的减速度
(单位:计数/毫秒², 浮点)



不能设为零!
(电机将不会减速)

电机 xx 的安全变量 (续)

- n lxx16: 运动程序里最大允许速度
(单位: 计数/毫秒 (cts/msec); 浮点)
- n 仅用于线性混合运动模式 (I13=0)
- n 进给率重载(百分比值)时比率有限
- n 如果I50=1, 如同快速(RAPID)模式下的指定速度

注意: 有些用户想提高运行运动程序时电机的运动速度, 但是忘记改变此变量, 结果电机速度会饱和在此设定处。

电机 x 的安全变量 (续)

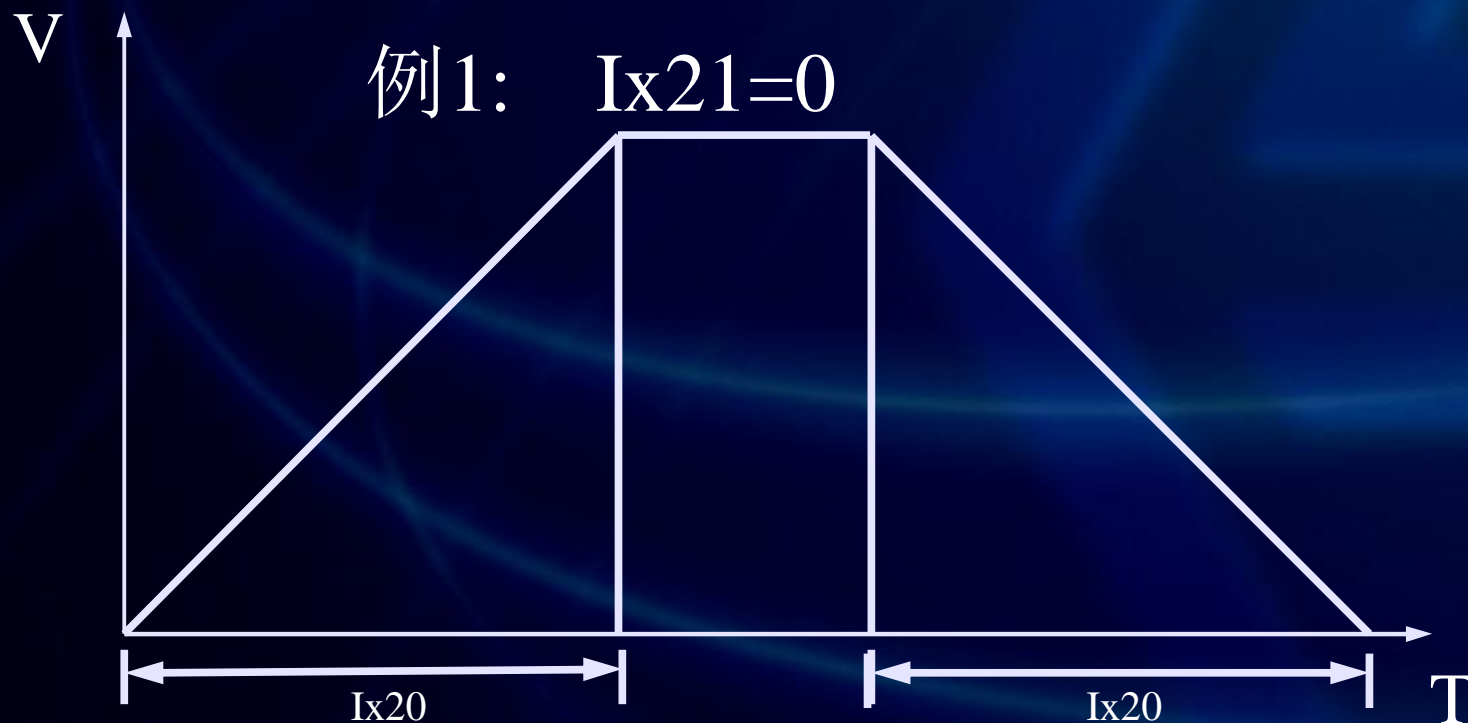
- n lxx17: 程序里允许的最大加速度
(单位: 计数/毫秒² (cts/msec²); 浮点)
- n 仅适用于线性混合运动模式(I13=0)
- n 进给率重载(百分比值)时比率有限
- n 当加减速需要不只一步规划的运动时, 效果有限

电机 xx 的安全变量 (续)

- n lxx19: 手动/回零/快速模式下最大允许加速度设置
(单位: 计数/毫秒² (cts/msec²); 浮点)
- n 能重载 TA (lxx20) 和 TS (lxx21) (与这两设置冲突时, 以此为准)
- n 若lxx20和 lxx21不为零时总使用它

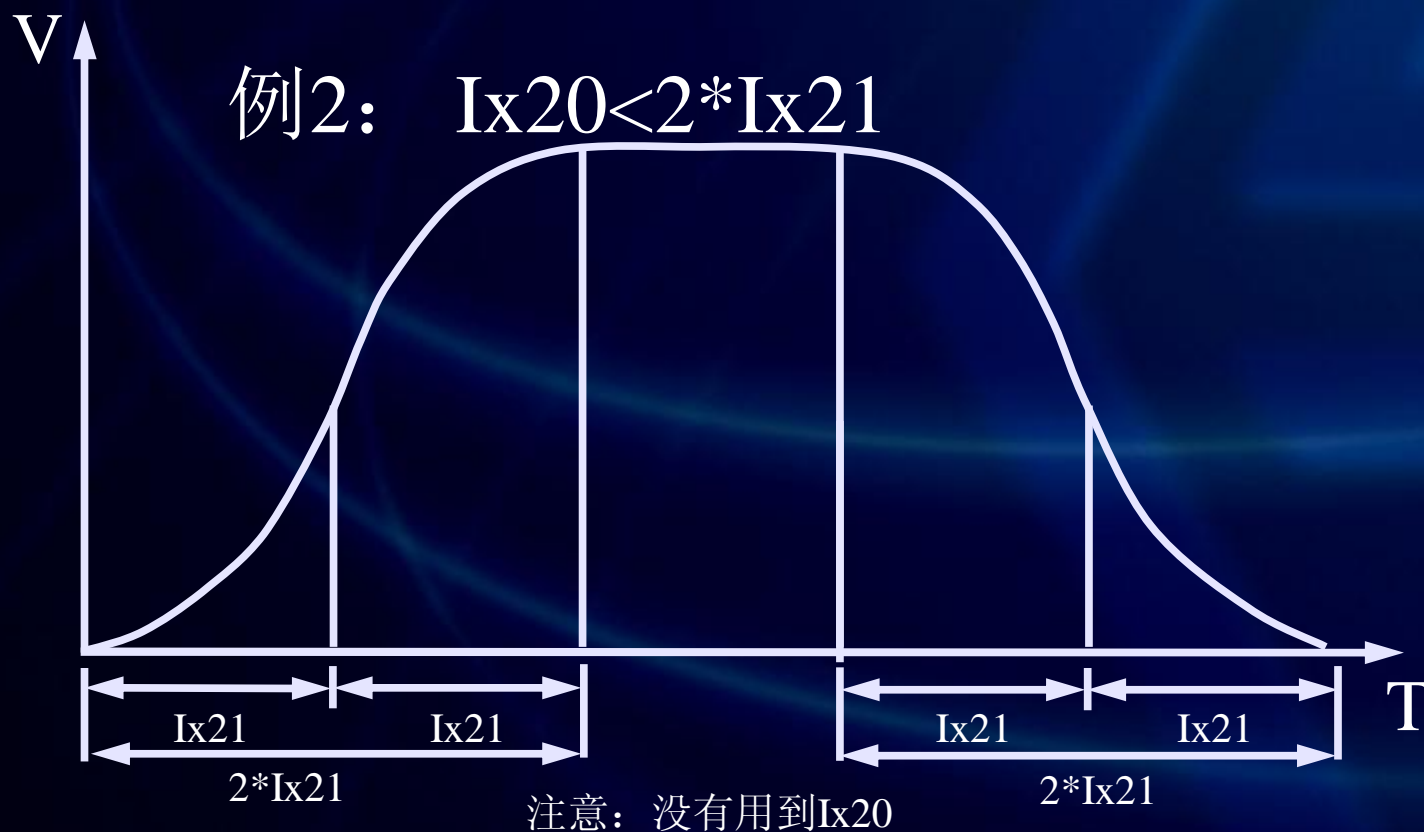
电机 xx 的运动变量

- n Ix20 加速时间 (手动Jog, 回零Home)设定
- n Ix21 S-曲线时间(Jog, Home)设定



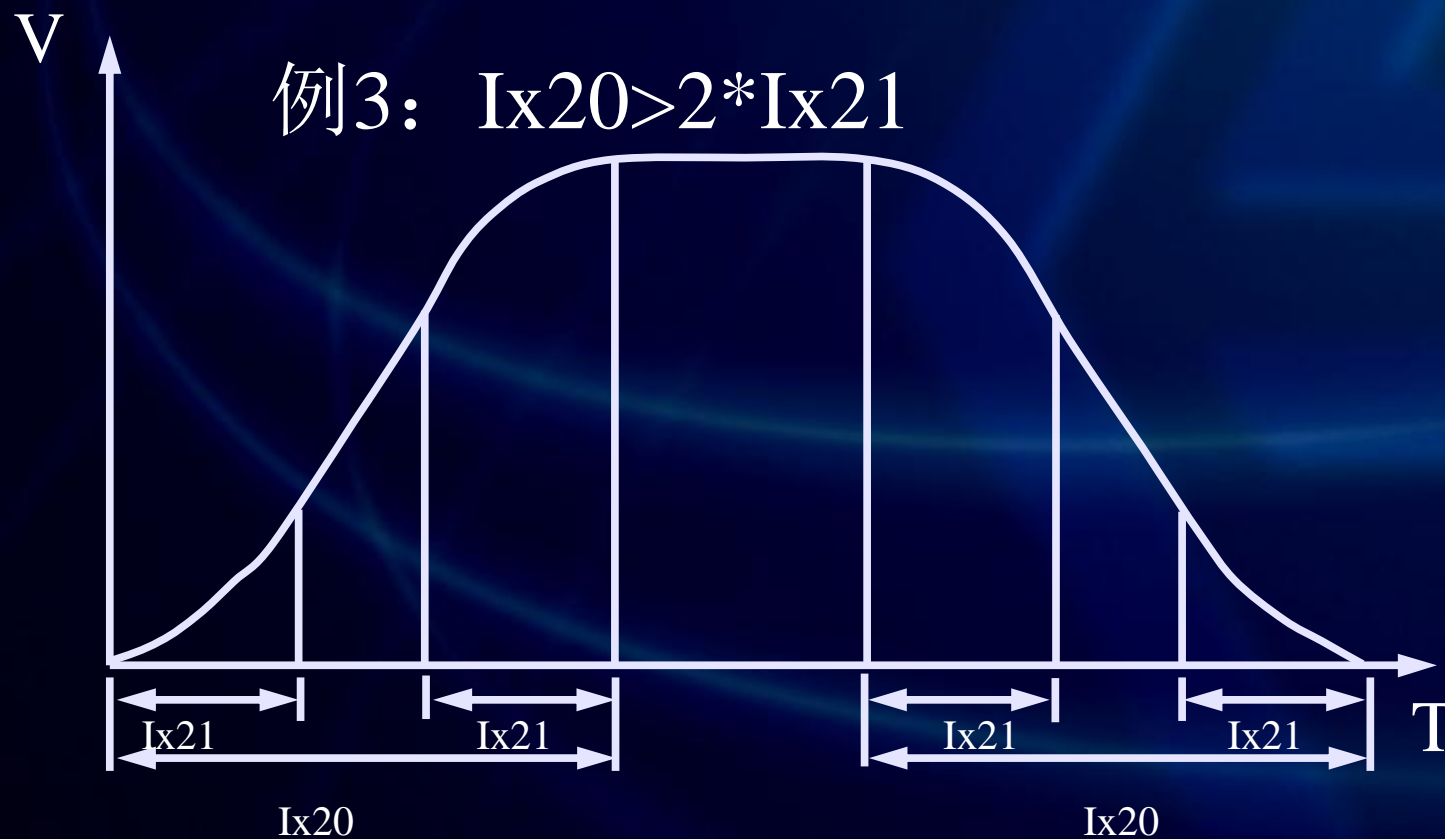
电机 x x的运动变量(续)

- n Ix20 加速时间 (Jog, Home)
- n Ix21 S-曲线时间(Jog, Home)



电机 xx 的运动变量(续)

- n Ix20 加速时间 (Jog, Home)
- n Ix21 S曲线时间 (Jog, Home)



电机 x 的运动变量(续)

- n lxx22: 手动速度 (绝对值)
(单位: cts/msec; 浮点)
- n lxx23: 回零速度 (符号表示方向)
(单位: cts/msec; 浮点)
- n lxx25: 限位/回零/故障/驱动器使能
[lxx25, lxx24] 标志地址和方式
(参看各章幻灯片)
- n lxx26: 回零偏移量 (单位: 1/16 ct)
回零触发(或者绝对值传感器零点)到
电机零位置的距离

电机 xx 的运动变量(续)

n lx27: 位置翻转范围 (单位: 编码器计数 (cts))
仅用于旋转轴(A,B,C)

若 $lx27 > 0$, 轴将采用绝对值方式进行较小距离运动 ($\leq lx27/2$ counts)

若 $lx27 < 0$, 目的为止符号表示方向, 幅值表示距离



ABS

...

A315

A45

- ① 无翻转
- ② 翻转

电机 xx 的运动变量(续)

- n lxx28: “在位置”范围 (单位: 1/16 ct)
仅用于指示目的
- n 电机被认为 “在位置” 如果满足下列条件:
 - n 闭环
 - n 期望速度为零
 - n 对应的运动控制程序的始终关闭f
 - n 跟随误差 < lxx28
 - n 所有上述条件在 (l7+1) [lxx88 +1] 个连续后台扫描周期内为真
- n 例子:

X10	;运动
DWELL0	;停止前瞻
WHILE(M140=0)WAIT	;是否“在位置”
M1=1	;“在位置”条件真并采取行动

上电时电机的绝对位置读取地址

- n lxx10: 上电伺服位置地址
控制PMAC在上电(或重启, 或发送\$*、\$\$*命令)时从绝对值位置传感器读取位置信息。
- n lxx95: 指定如何解释上电时读取的绝对位置信息(如果存在的话)
 - n 可能的格式:
 - n Acc-8D Opt 7 R/D 转换器
 - n Y-寄存器并行数据
 - n Acc-28 数模转换结果
 - n Acc-49 三洋绝对值编码器
 - n X-寄存器并行数据
 - n Acc-8D Opt 9安川绝对值编码器.
 - n MACRO Station安川绝对值编码器.
 - n MACRO Station R/D 转换器
 - n MACRO Station 并行读取

PMAC 手动(Jog) 命令

基本的手动命令

n J+	手动正向运行命令
n J-	手动负向运行命令
n J/	手动停止 (或闭环)
n J=	手动返回到上次手动(上一程序)的位置
n J={常数}	手动运行到指定位置 (单位: 编码器计数)
n J^{常数}	从当前实际位置手动运行指定距离
n J:{常数}	从当前命令的位置手动到指定距离

变量 **lx19 - lx22**控制手动。这些变量可以在运行时修改，但直到下一个手动命令开始才起作用。

与寄存器变量相关的手动命令

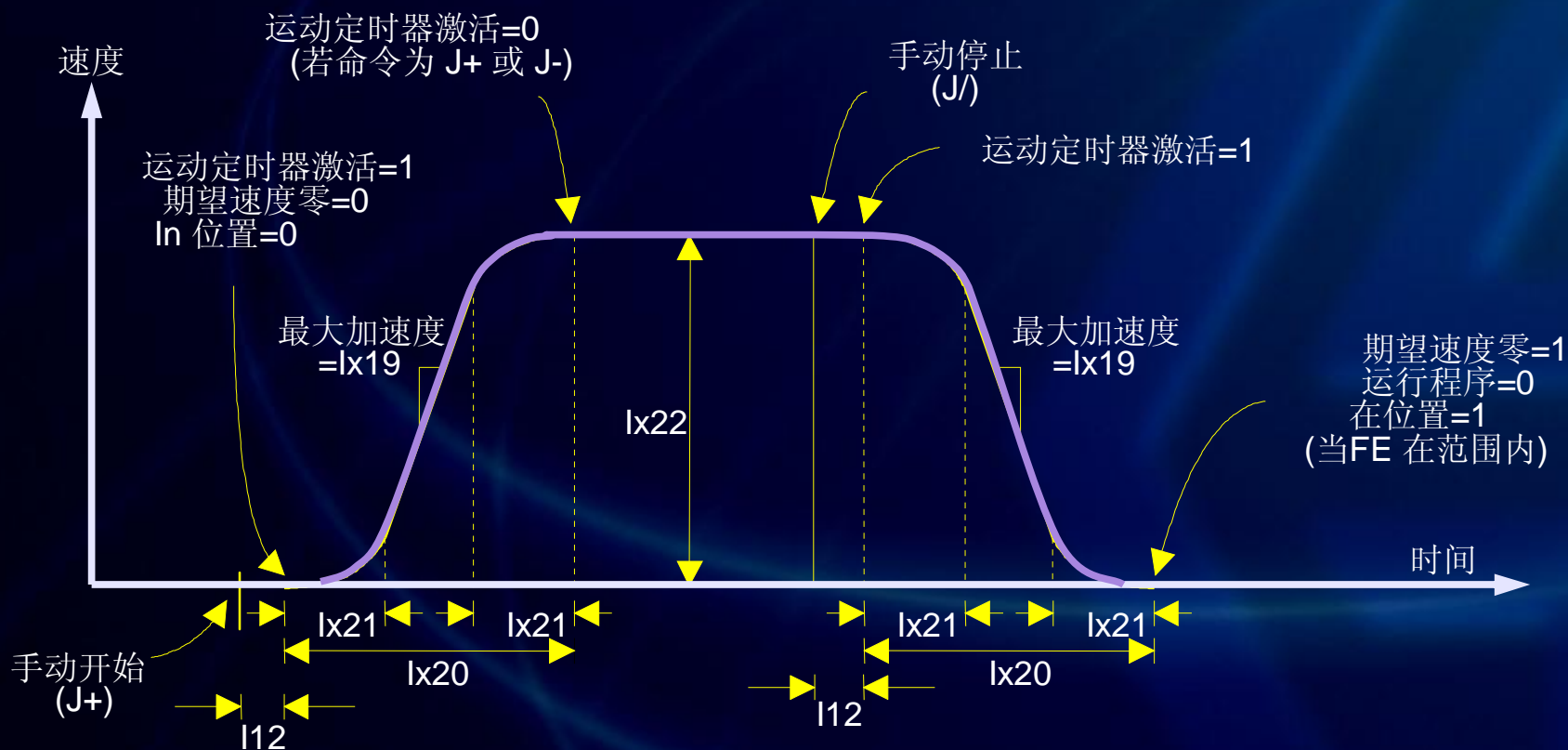
- n** $J=^*$ 手动到手动变量寄存器指定的位置 (in counts)
- n** $J\wedge^*$ 手动到手动变量寄存器指定的距离(从当前实际位置)
- n** $J:^*$ 手动到手动变量寄存器指定的距离(从当前指令位置)
- n** $Mx72$ 为用于指定手动变量寄存器的M-变量

n 例子:

$$M172 = P1 + 500$$

$$\#1J=^*$$

手动运动轨迹



注：如果手动命令不是 无限运动
(非 J+ 或 J-) 那么“运行运动”
(“Running a Program”) 状态位
将在整个手动过程中被置为高

注：lx19用来限制加速率，它
可被变量lx20和lx21重载

手动 命令实例操作

- n 试一些这个章节里讲到的手动命令
- n 试改变lxx19, lxx20, lxx21和lxx22的设置，观察这些变量对手动的影响。

PMAC 触发式手动运动 (回零运动)

PMAC 的触发式运动

- n 这种运动与触发-捕获位置相关
- n 三类触发运动:
 - n - 回零寻找运动
 - n - 在线手动直到触发
 - n - 运动控制程序里运动直到触发 (RAPID模式)
- n 两类触发:
 - n - 输入触发: 索引和/或标志
 - n - 误差触发: 超越警告跟随误差
- n 两种位置捕获方法:
 - n - 硬件捕获: 立即的; 只与编码计数器计数有关
 - n - 软件捕获: 最多1个后台周期的延时

触发式运动

输入触发 vs. 误差触发

n 输入触发 (标志和/或索引边沿)

- n - lxx97 位1 = 0 (缺省) 选择
- n - lxx25 指定索引或标志位地址
- n - l7mn2 选择使用标志或索引, 或者设定它们的极性
- n - l7mn3 选择使用哪一个标志
- n - “捕获”标志状态位 = 1 表示触发

n 误差触发 (警告跟随误差-WFE)

- n - lxx97 位1 = 1 选择
- n - lx12 设置WFE限制的幅值
- n - 必须设置“软件捕获” lxx97 的0位 = 1
- n - 电机WFE的状态位 = 1 表示触发

触发式运动

硬件捕获 vs. 软件捕获

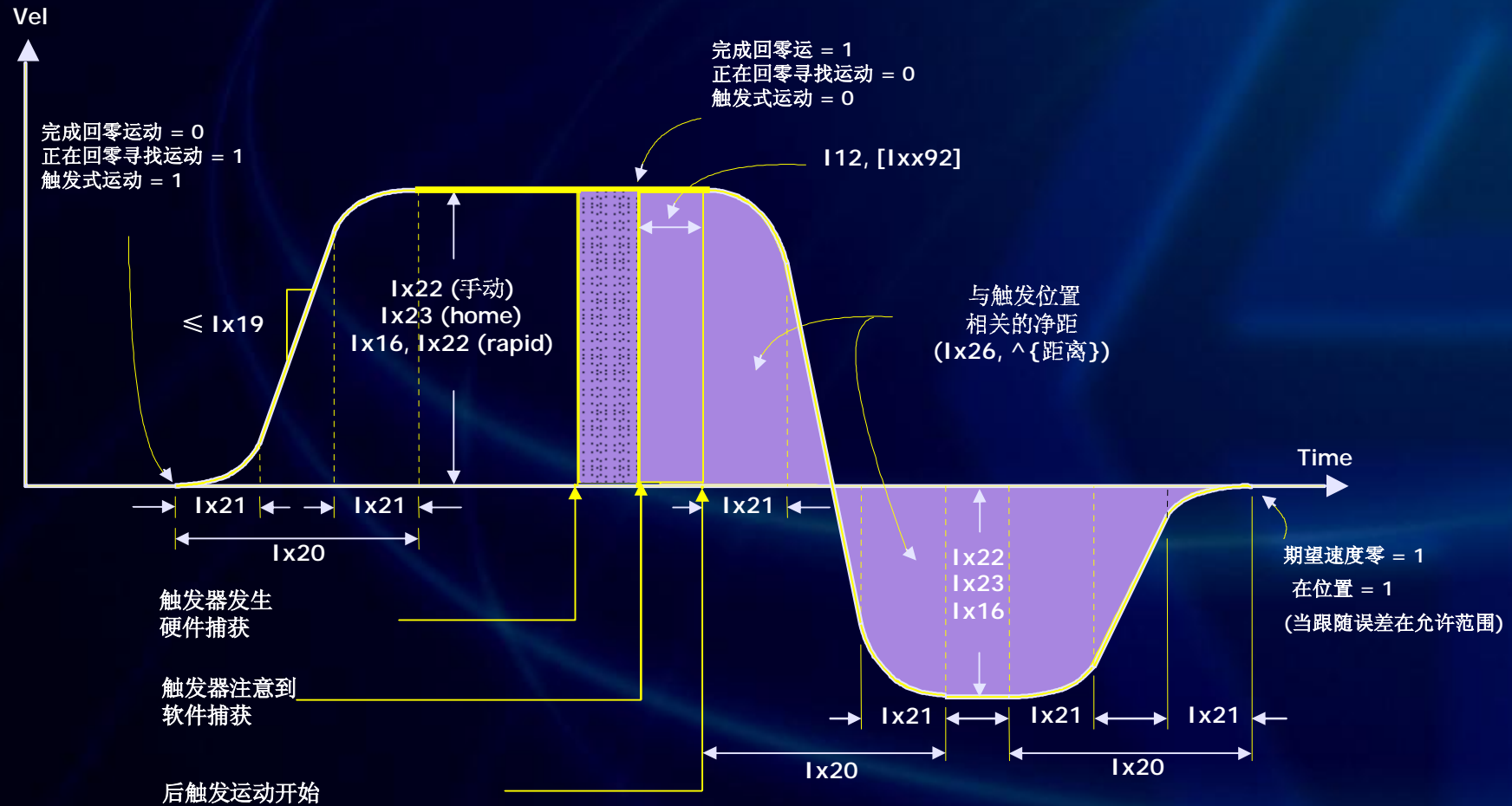
n 硬件捕获

- n - lxx97 位 0 = 0 (缺省)选择
- n - 要求为输入触发，不是误差触发
- n - 要求通过编码计数器得到位置反馈
- n - 要求同一硬件通道得到位置和标志
- n - 任何速度下捕获得到确切的位置计数

n 软件捕获

- n - lxx97 位0 = 1选择
- n - 允许输入触发或者误差触发
- n - 任意类型的位置反馈
- n - 可能存在正比于速度的捕获误差

触发式运动轨迹



回零运动实例操作

- n 在演示软件中，根据下列条件设置电机回零
 - n 编码器索引脉冲
 - n 回零开关
 - n 警告跟随误差

PMAC 伺服参数整定(调节)

PMAC 整定

- n 对任何包括电机和伺服器的系统，必须配置好PMAC的伺服算法已达到正确的运动控制效果。
- n 配置参数即为调整与“PID增益”相关的I-变量
- n 确定合适的PID参数的过程称为“整定”

典型的 P.I.D 伺服环



Ixx30

P (比例增益)

Ixx33

I (积分增益)

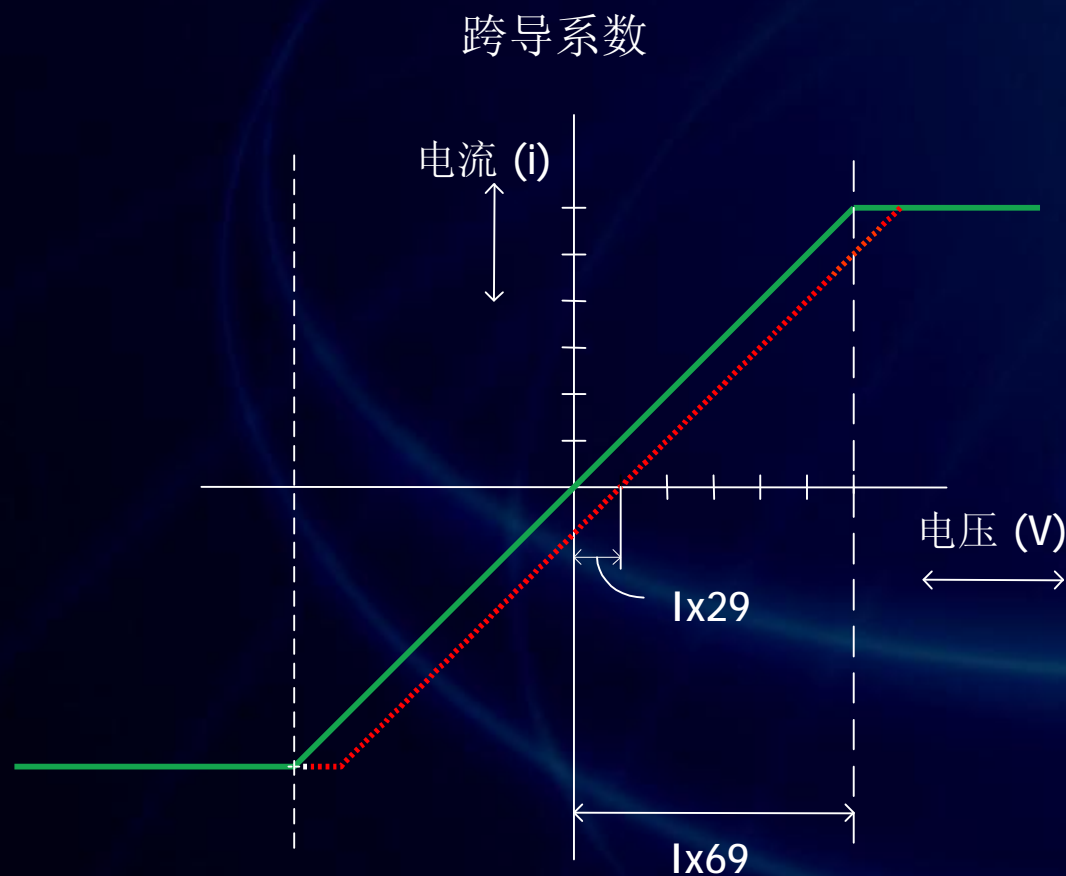
Ixx31

D (微分增益)

整定参数

n	lx30	比例增益
n	lx31	微分增益
n	lx32	速度前馈增益
n	lx33	积分增益
n	lx34	积分方式
n	lx35	加速度前馈增益
n	lx68	摩擦前馈增益
n	lx69	最大输出设置
n	lx29	数模变换输出(DAC)的偏移量

设置Ixx69和Ixx29

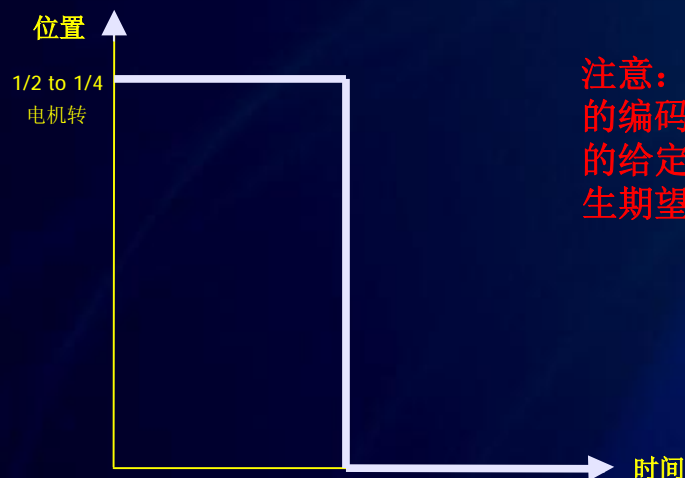


Ixx69的设置16位DAC中：
取值范围为1- 32,767。如果
伺服器的线性输出范围为
10V，则Ixx69的缺省值
20,480对应于6.25V的最大
输出。

Ixx29 用于调节电机 xx 的输
出偏差

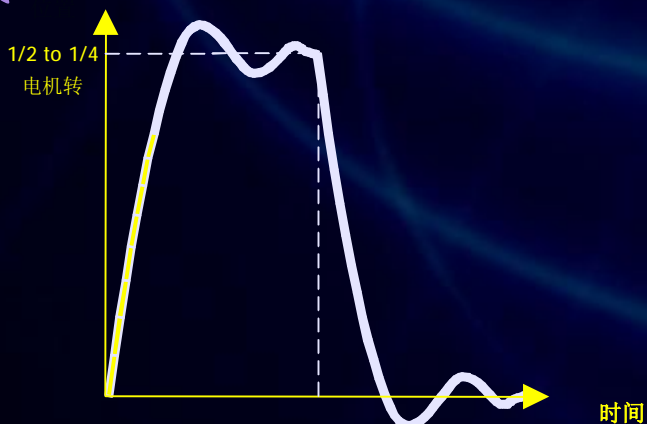
阶跃运动整定

指令曲线:



注意: 做伺服环整定时, 请根据每转对应的编码器或反馈装置的计数设置阶跃指令的给定量, 缺省值有时相对太小不足以产生期望的运动

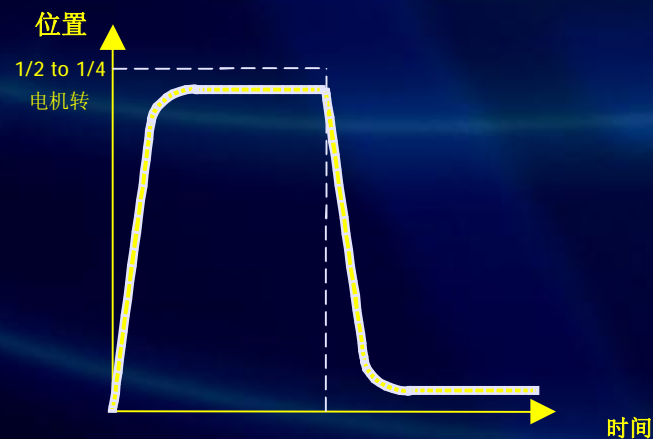
实际响应曲线
和调节建议:



超调和振荡

原因: 阻尼太小或者比例增益太大

解决: 减小 K_p (Ix31)
增大 K_d (Ix30)



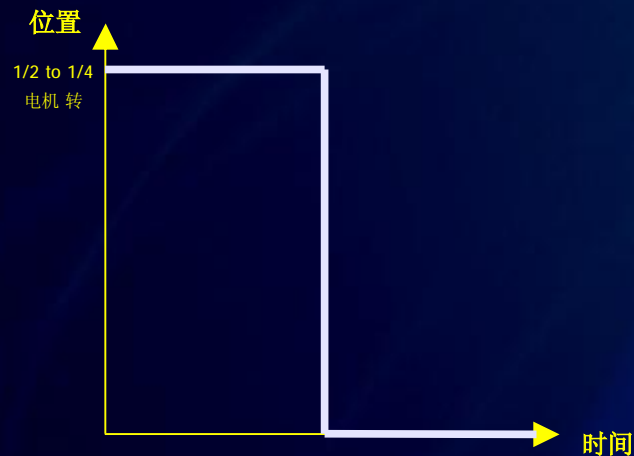
位置净差

原因: 摩擦或者恒定阻力

解决: 加大 K_i (Ix33)

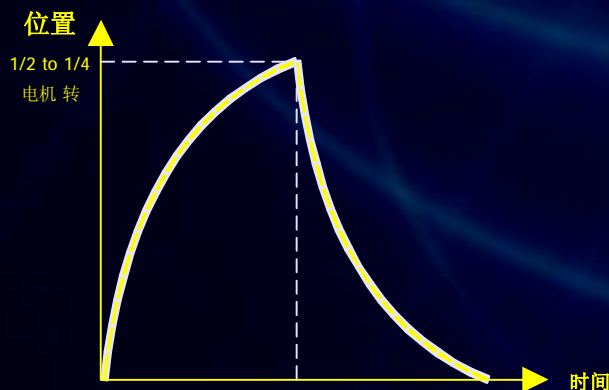
阶跃运动整定(续)

指令曲线:



指令位置

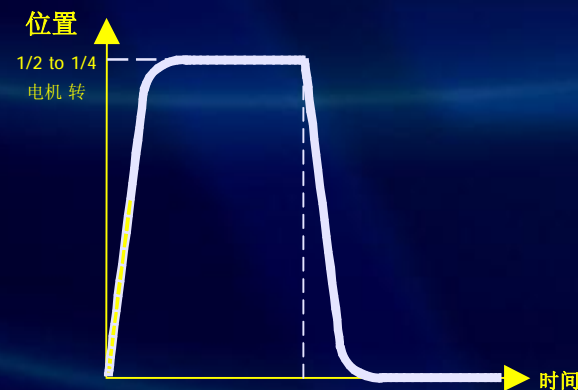
实际响应曲线
和调节建议:



响应迟缓

原因: 阻尼太大或者比例增益太小

解决: 增大 K_p (Ix30) 或者
减小 K_d (Ix31)



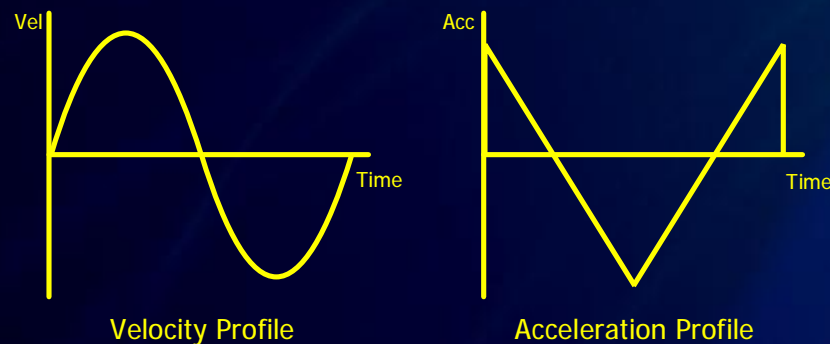
实物系统的限制

原因: 电机/伺服器/负载的物理限制

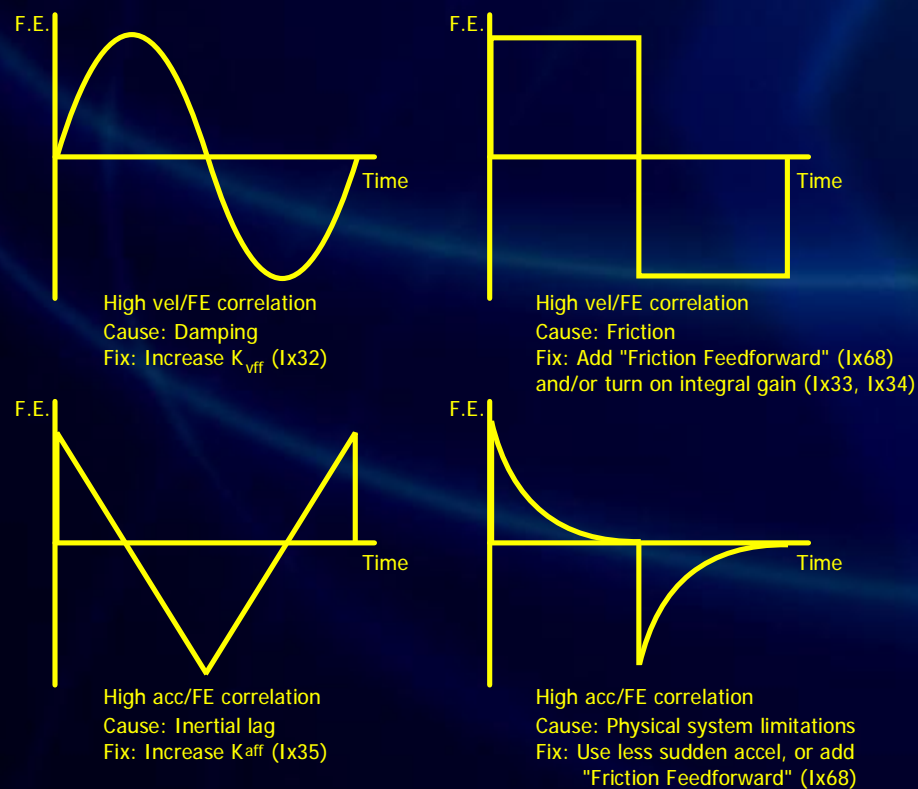
解决: 评估性能, 也许可以增大一点 K_i (Ix30)

二次曲线运动整定

指令曲线:

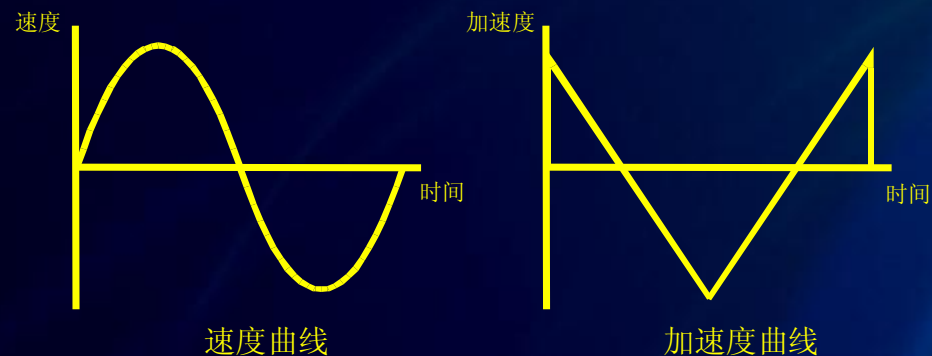


实际响应曲线
和调节建议:

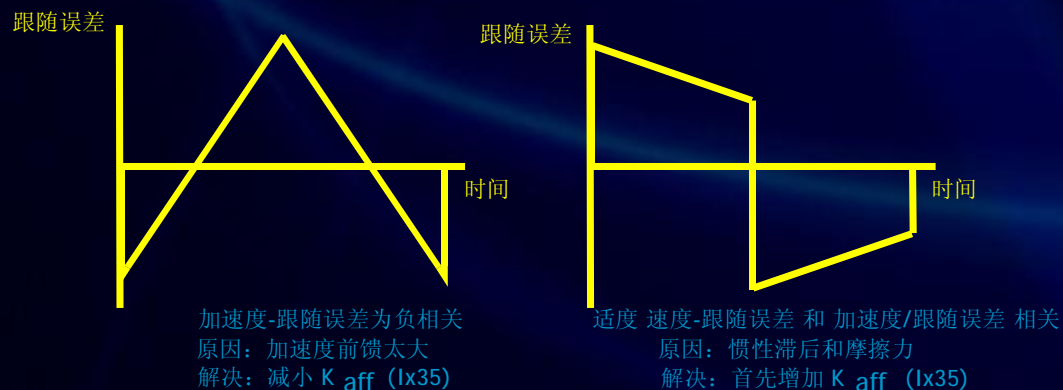
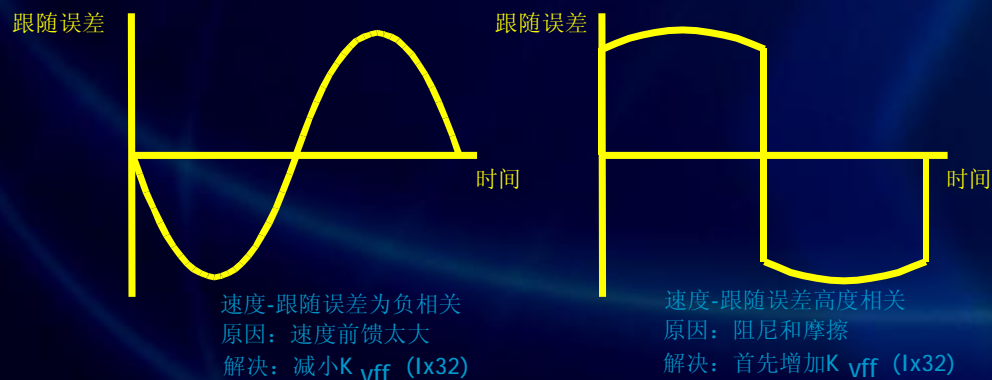


二次曲线运动整定(续)

指令曲线:



实际响应曲线
和调节建议:



PMAC Tuning Pro2软件

- n PEWIN32PRO2软件内装有电机整定的实用工具，叫做PMAC Tuning Pro2软件，它在Tools的下拉菜单中。
- n PMAC Tuning Pro2软件的最大特点是：带有著名的、强大的自整定程序，提供了快速且简便的方法以分析和整定电机驱动系统。



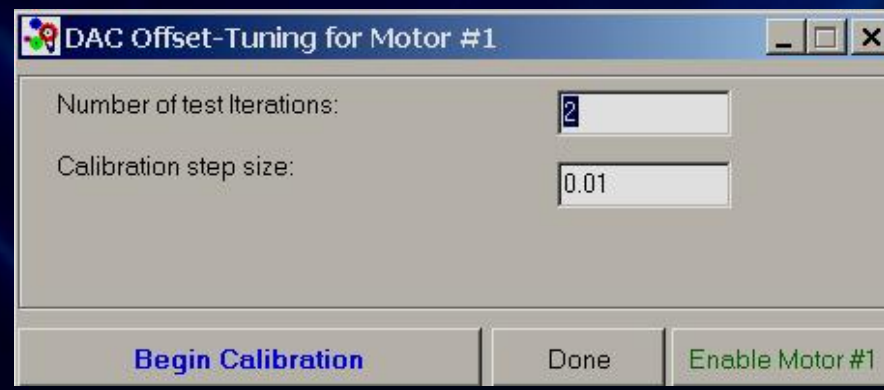
PMAC的自整定

- n 自整定功能适用于从未使用过PMAC的用户。 出于安全考虑，对新客户建议使电机脱离负载来熟悉自动整定功能。一旦掌握之后可以带负载调整。
- n 自动整定将快速运动电机并根据动态响应计算电机的 PID 增益。如果在自整定器计算增益时在带负载的情况下不能使系统产生期望的运动，说明不能自整定带负载的电机。这时，你必须手动整定电机伺服环参数。

PMAC的自动整定过程

校正DAC:

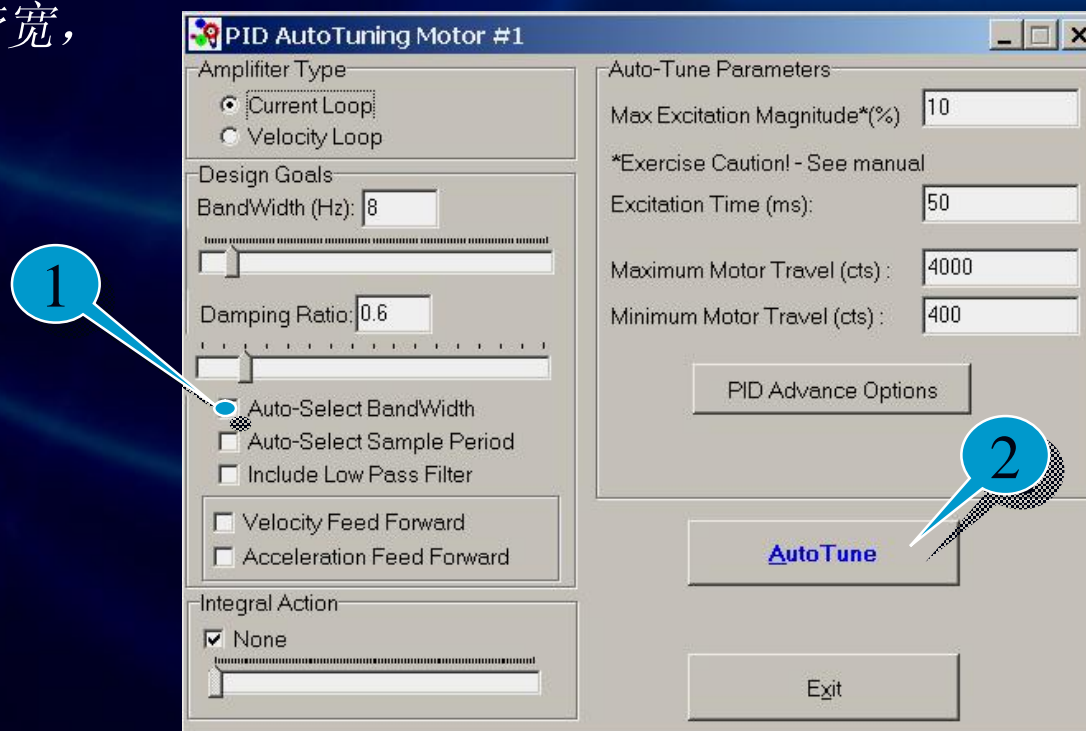
使用Tuning Pro中的自动DAC校正，或者手动调节伺服器上的电位器，保证0%指令输出时电机不动。



PMAC的自动整定过程(续)

- n 使用自整定
- n 选择“Auto-Select Bandwidth”，确定没有勾选“Velocity Feed Forward”，“Acceleration Feed Forward”，和“Integral Action”项，然后点击“Auto Tune”按钮。

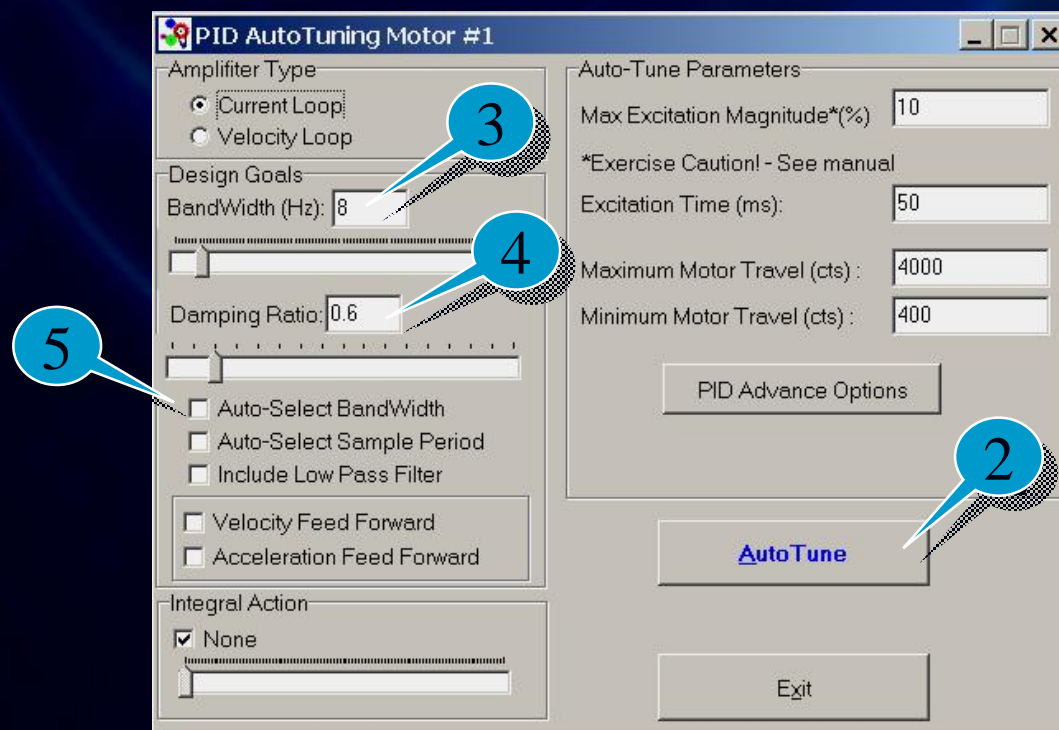
这一过程将计算出一个保守带宽，比例增益和微分增益值。



PMAC的自动整定过程(续)

创建PID控制器 (阶跃运动)

- n 利用自整定得到的保守带宽，将其乘以2或3，然后输入到带宽项
- n 设定阻尼比为0.7 (大部分系统可以接受)
- n 不勾选“Auto Select Bandwidth”项，然后按“Re-calculate Gains”按钮
- n 做阶跃运动，重新计算增益



PMAC的自动整定过程(续)

创建PID前馈控制器 (二次曲线运动)

- n 选择软件或者硬件积分方式 (用户可配置, 取决于系统)
- n 选择速度前馈, 加速度前馈, 然后选择重新计算。
- n 增加带宽, 然后点击 'Recalculate' 按钮。
 - n 不断增加带宽, 直到电机振动/蜂鸣
 - n 稍微减小带宽, 直到电机不再振动

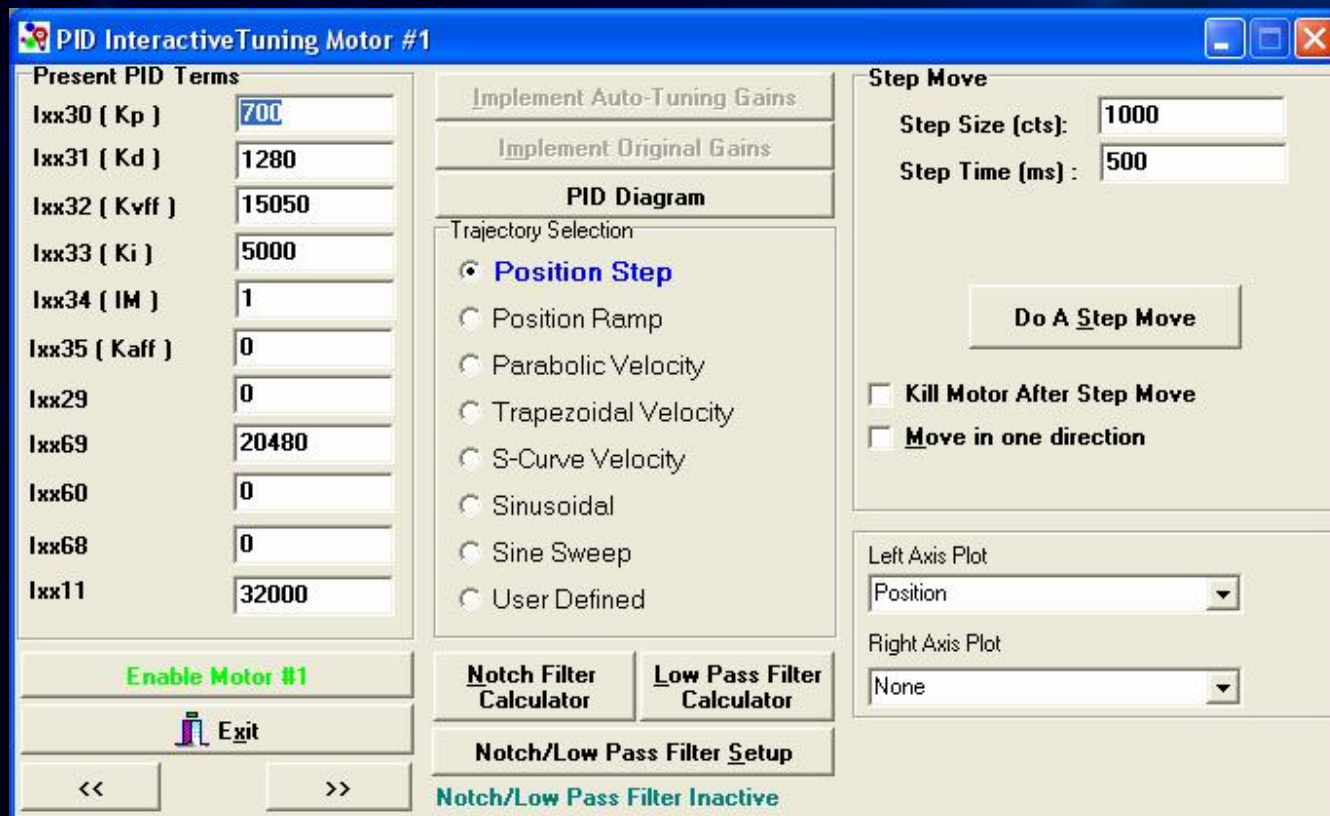
电机交互式整定

为了使交互整定具有很好的整定性能:

首先选择“Position Step”响应

然后选择“Parabolic Velocity”响应

- n 加上速度前馈 (二次曲线运动)
- n 加上加速度前馈 (二次曲线运动)

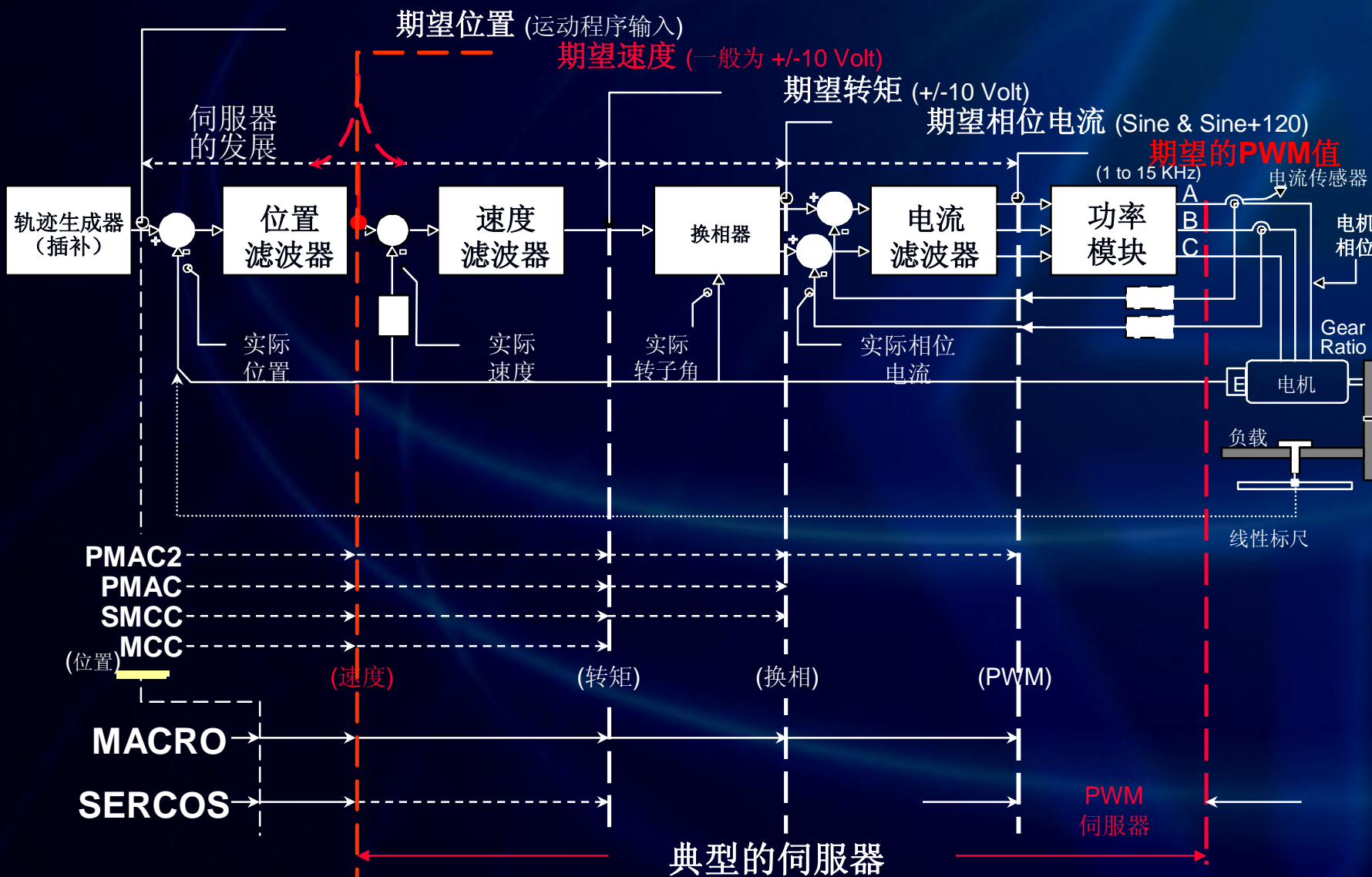


电机整定的练习

- n 在终端窗口，设置所有的Ixx30..xx39 变量为0值。这样将清除电机 xx 伺服环的设定值。
 - n 键入命令：I130..139=0
- n 从“Tools”下拉菜单中启动PMAC Tuning Pro2。调整所有的PMAC电机使跟随误差最小。

控制器/ 伺服器接口的发展

闭环伺服 (控制器-伺服器接口的发展)



速度模式伺服器

- n 伺服器内速度环和电流环闭环
- n 在电机 (有刷直流)或伺服器(无刷)内进行换相
- n 速度环参数取决于负载
 - n 必须由机器制造商设定
- n 通常采用模拟量进行速度环闭环
 - n 增益参数不受数字限制
 - n 应用时增益参数不容易调节

转矩模式伺服器

- n 伺服器内电流环闭环
- n 电机 (有刷直流)或者伺服器(无刷)内进行换相
- n 控制器内的速度环必须闭环
 - n 受数字系统限制
- n 伺服器独立于负载
 - n 可以由伺服器制造商进行设定

正弦波输入伺服器

- n 速度环和换相都在控制器(例如PMAC)内完成
- n 相位电流有两个模拟电压命令输出
- n 伺服器内相位电流环闭环
- n 几乎都是模拟电流环
- n 通常为线性调制的功率晶体管

功率模块伺服器

- n 速度环、换相和电流环都在控制器(例如PMAC)内完成
- n 3相电压命令输出到伺服器，形成控制上&下桥臂的3个PWM对
- n 功率晶体管实际接收到开/关命令
- n 相电流以串行格式反馈到控制器

PMAC 换相

正弦换相
直接PWM控制

为什么需要换相？

- n 电机要求在相的AC电流
- n 能有效的从DC源控制电机
- n 换相改变电机相位的方向，以保持期望的转矩方向
- n 也可以控制电机相位中的电流幅值，以保持期望的转矩值
- n 有刷电机机械换相
- n 无刷电机电子换相

为什么采用PMAC换相？

- n 不需要对驱动器进行位置反馈
- n 驱动器将更加通用并且造价低
- n 具有优质的换相算法
- n 同步电机从增量式编码器换相
- n 感抗电机采用矢量伺服驱动
- n 反馈丢失或反馈搞反只会导致失去转矩, 不会导致飞车

电机电枢和励磁

n 电枢

- n 大电流（快速变化）
- n 在有刷电机的转子上
- n 在无刷电机的定子上
- n 在交流电机各自的相绕组上

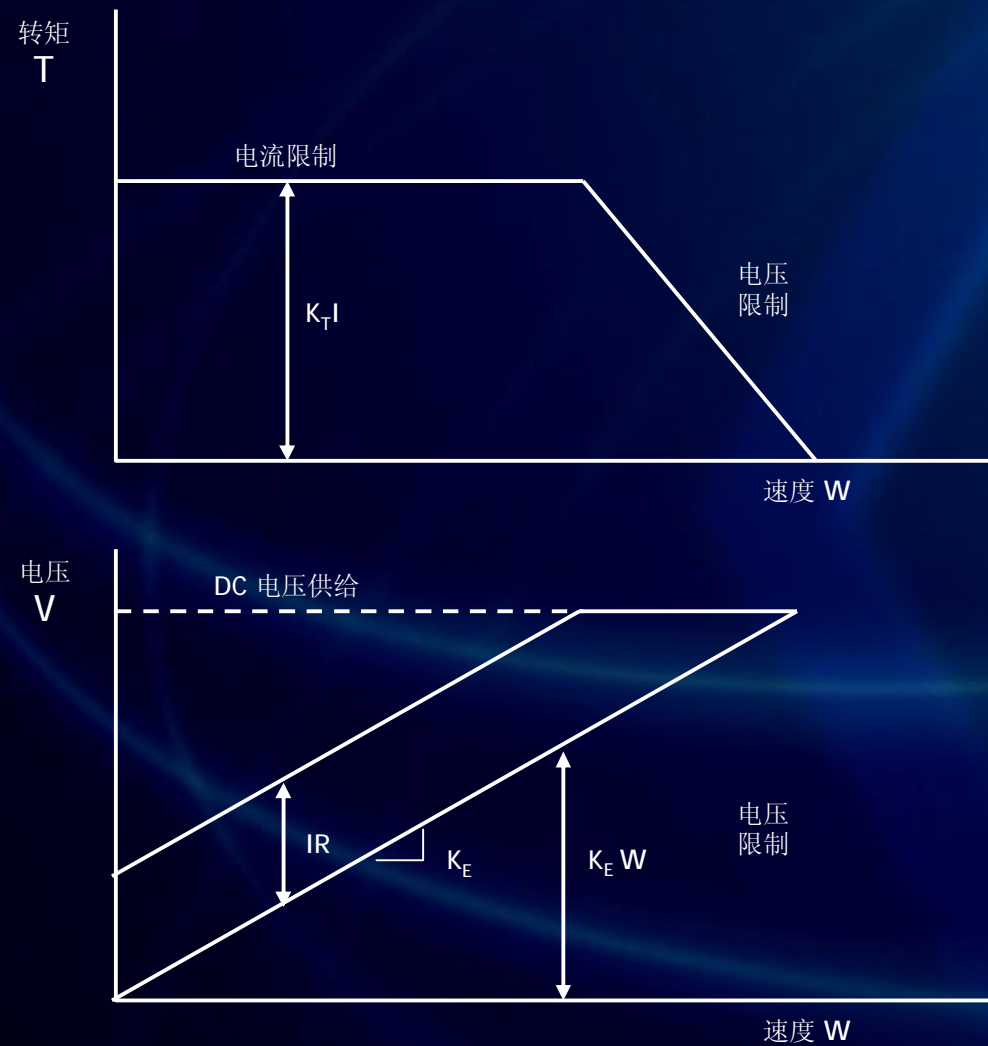
n 励磁绕组

- n 小电流（连续变化，不变或者慢速变化）
- n 在有刷电机的定子上
- n 在无刷电机的转子上

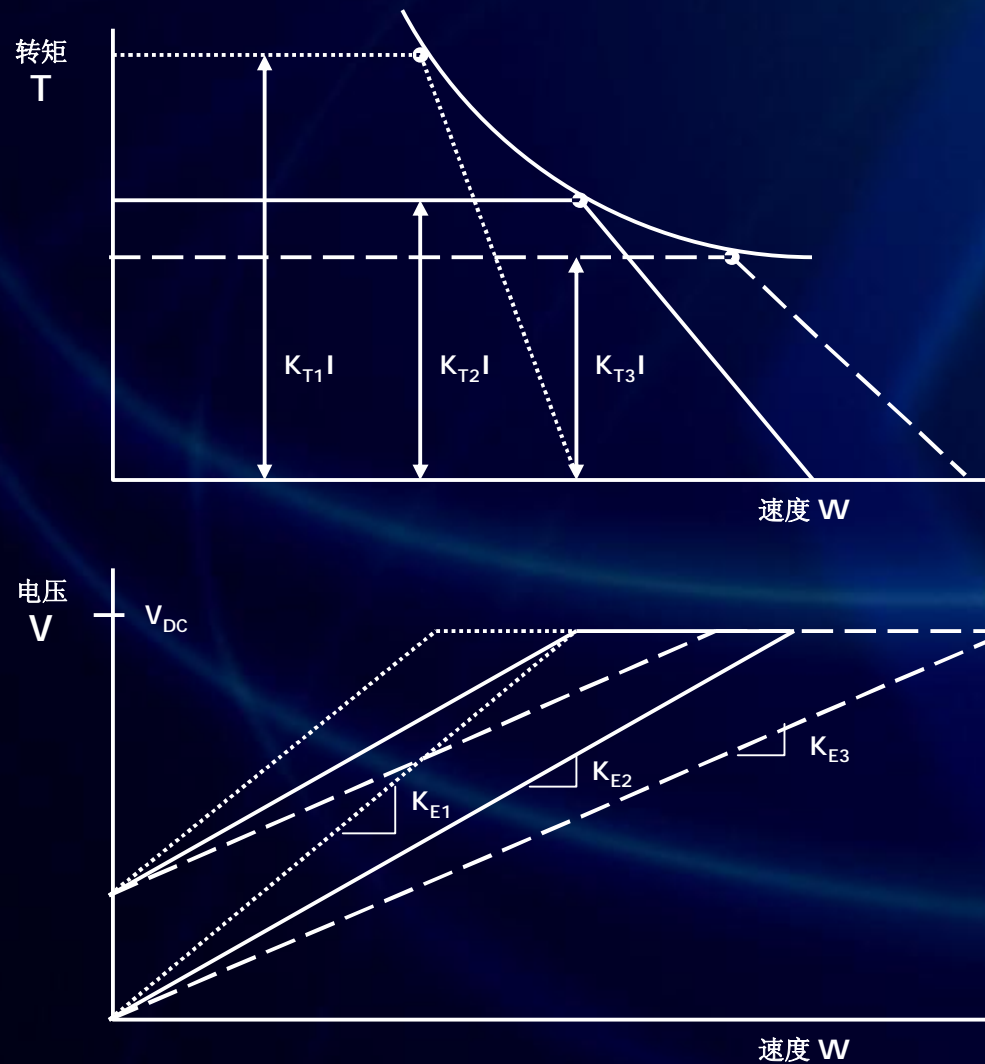
换相: 转子励磁控制

- n 两种产生转子电磁场的方法
 - n 永磁体: 磁体内部电子产生磁场
 - n 电磁体: 电流产生磁场
- n 转子电磁场强度 (λR) 决定:
 - n 转矩常数 K_T : ($T = K_T I A$)
 - n 反电动势常数 K_E : ($E = K_E \omega$)
(注: $E I = T \omega$, 因此 $K_T = K_E$)
- n 电磁体的转子电磁场强度会动态的变化
 - n 大转矩在低速时呈现强磁场
 - n 低的反电动势在高速时呈现弱磁场
 - n 这一技术称为“磁场弱化”
 - n 创建“软调节”

电机转矩性能 — 一定励磁



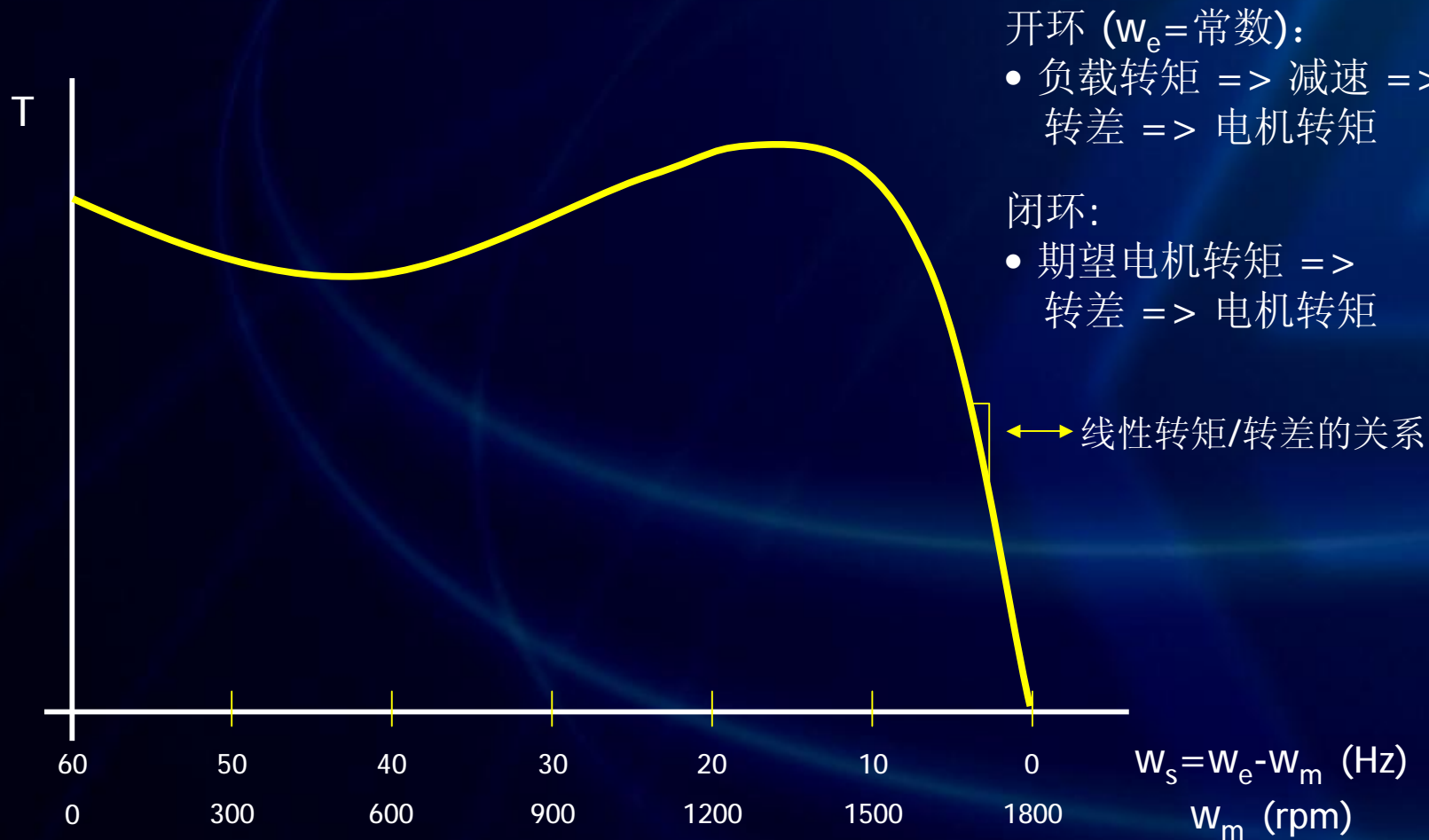
电机转矩性能－变励磁



什么是矢量控制？

- n 感抗电机应用的换相算法使其具有伺服电机的定位能力
- n AKA “磁通矢量控制”，“磁场定向控制”
- n 在永磁无刷电机中有两点未呈现：
 - n 必须产生转子电流从而生成电磁场
 - n 必须创建转子磁场和转子之间的“转差”
- n “磁化电流”参数可动态地控制转子磁场强度
- n “转差率”参数的设置必须与电机的动态性能匹配

感抗电机运行原理



为什么使用矢量控制？

- n 感抗电机的性价比很高，特别是在大功率时
- n 感抗电机功率可以做的很大
- n 感抗电机很健固
- n 感抗电机不存在去磁的可能性
- n 此技术可以消除飞车的可能
- n 磁场强度控制允许有灵活的转矩/速度范围

PMAC的换相算法

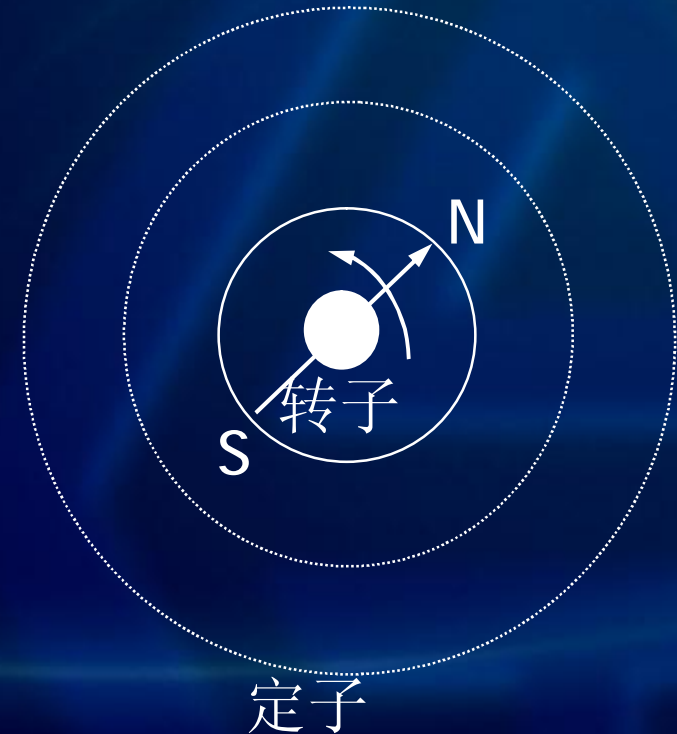
I. 计算转子磁场定向

A. 同步电机

1. 读取编码器位置
2. 对比参考位置
(上电或绝对)

B. 异步电机

1. 读取编码器位置
2. 计算转差频率
3. 在转子位置前加转差



PMAC的换相算法(续)

II. 对齐定子电流

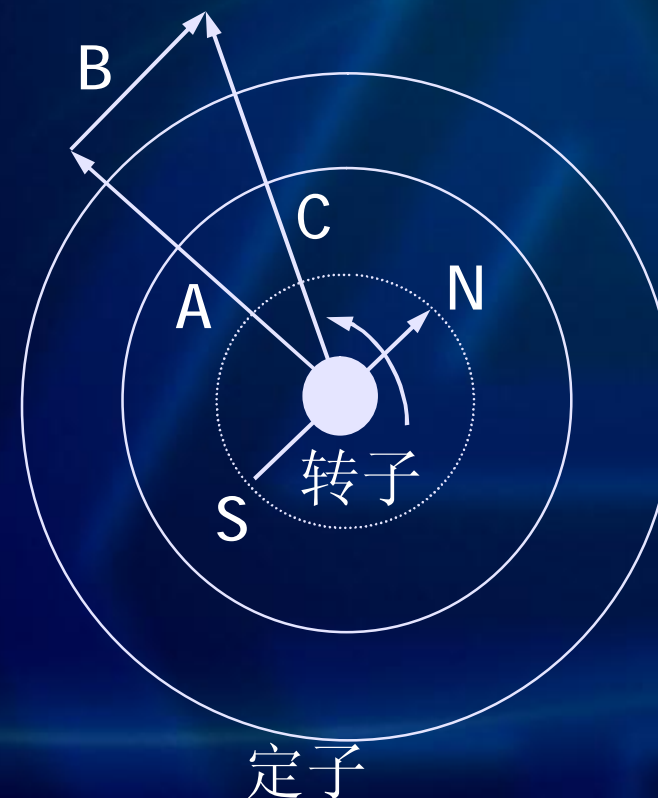
A. 转矩指令 (“正交”) 电流

其幅值由伺服控制/滤波器或开环产生
与转子磁场垂直

B. 磁化 (“直接”) 电流

主要用于感抗电机
与转子磁场平行

C. 两个成分采用向量加法



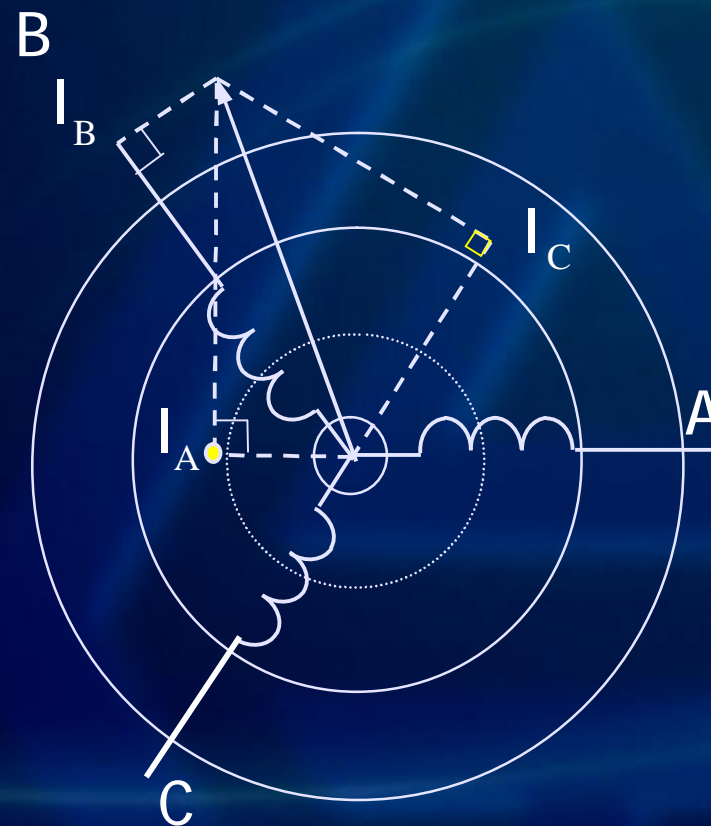
PMAC 换相算法(续)

III. 将电流与相位对齐

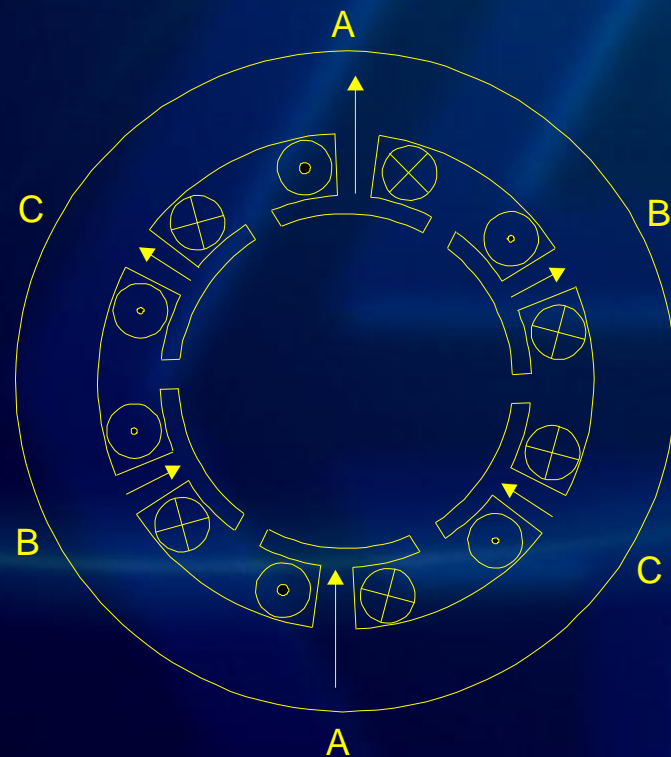
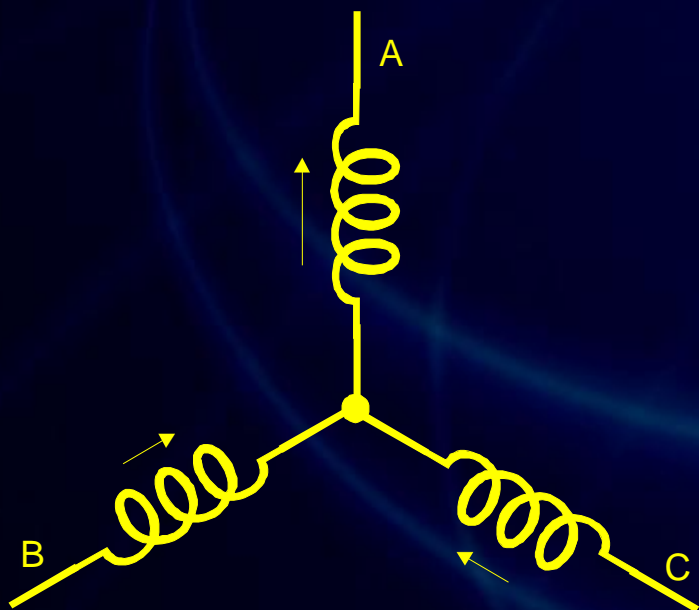
A. 从图形上说, 将电流矢量分解到各相

B. 算术上, 将电流矢量和每一单位
相位矢量进行点乘

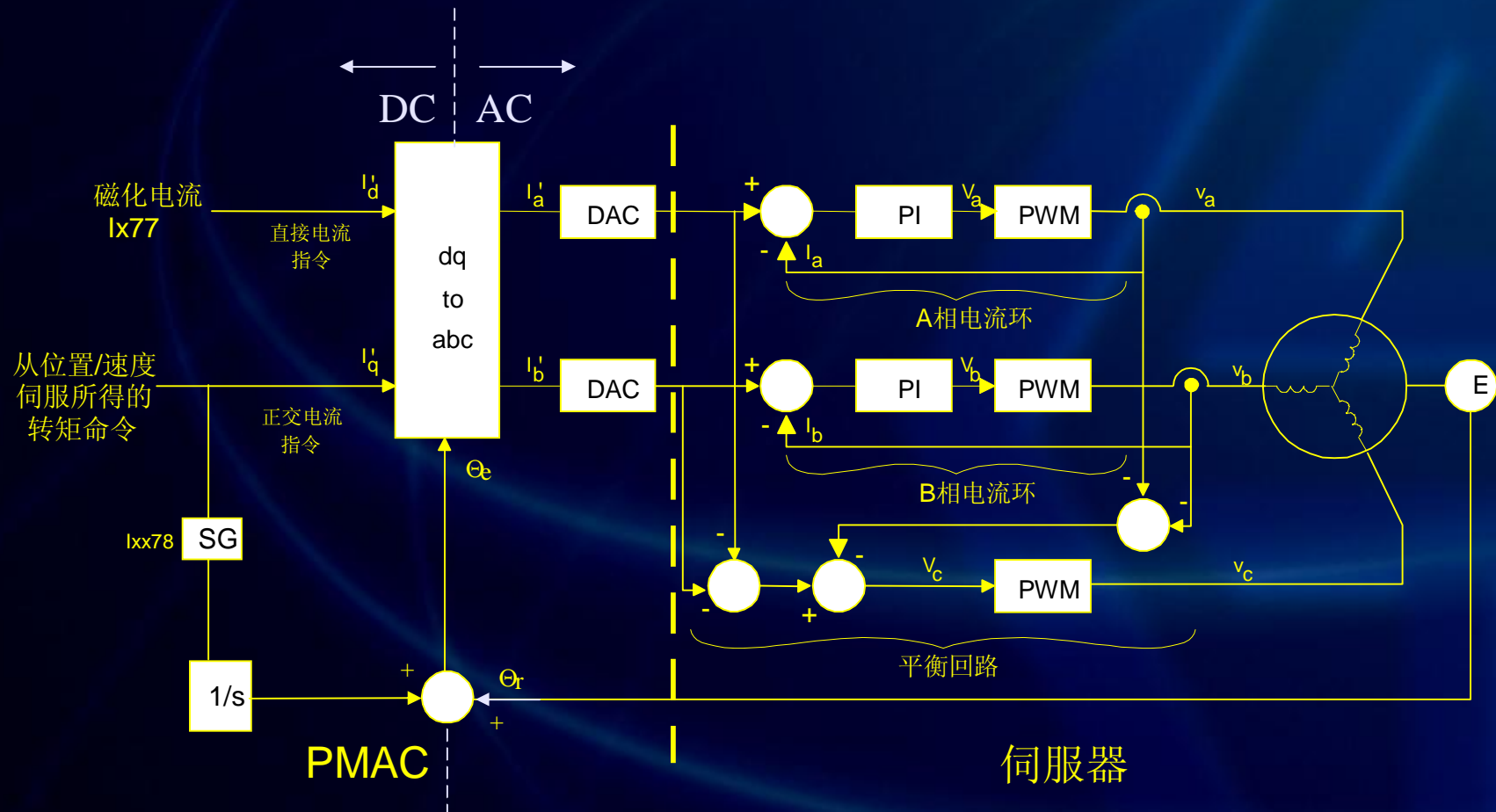
C. 只需计算2个相位; 平衡回路会自己产生第3(和第4) 相



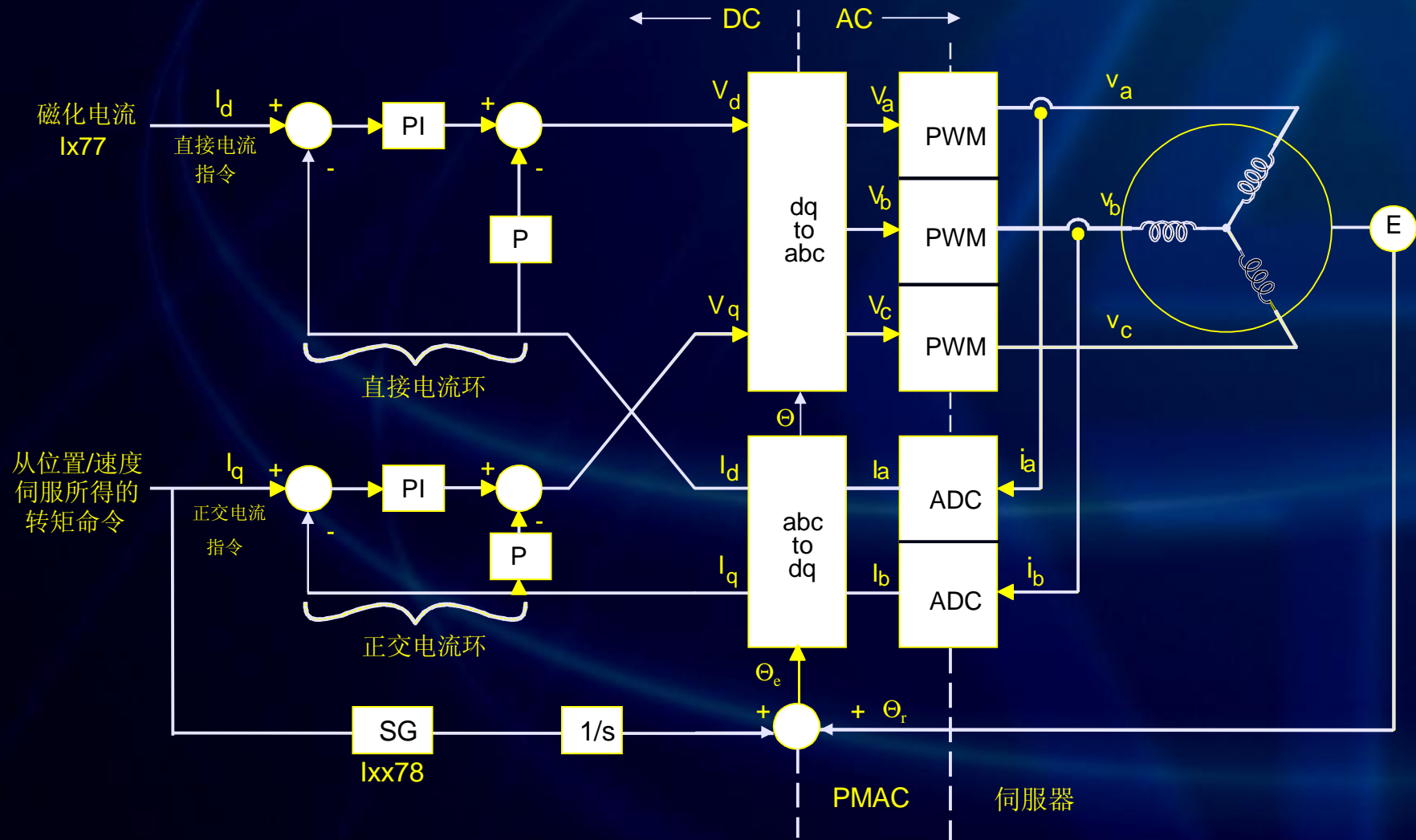
实际电机绕圈的绕组图关系



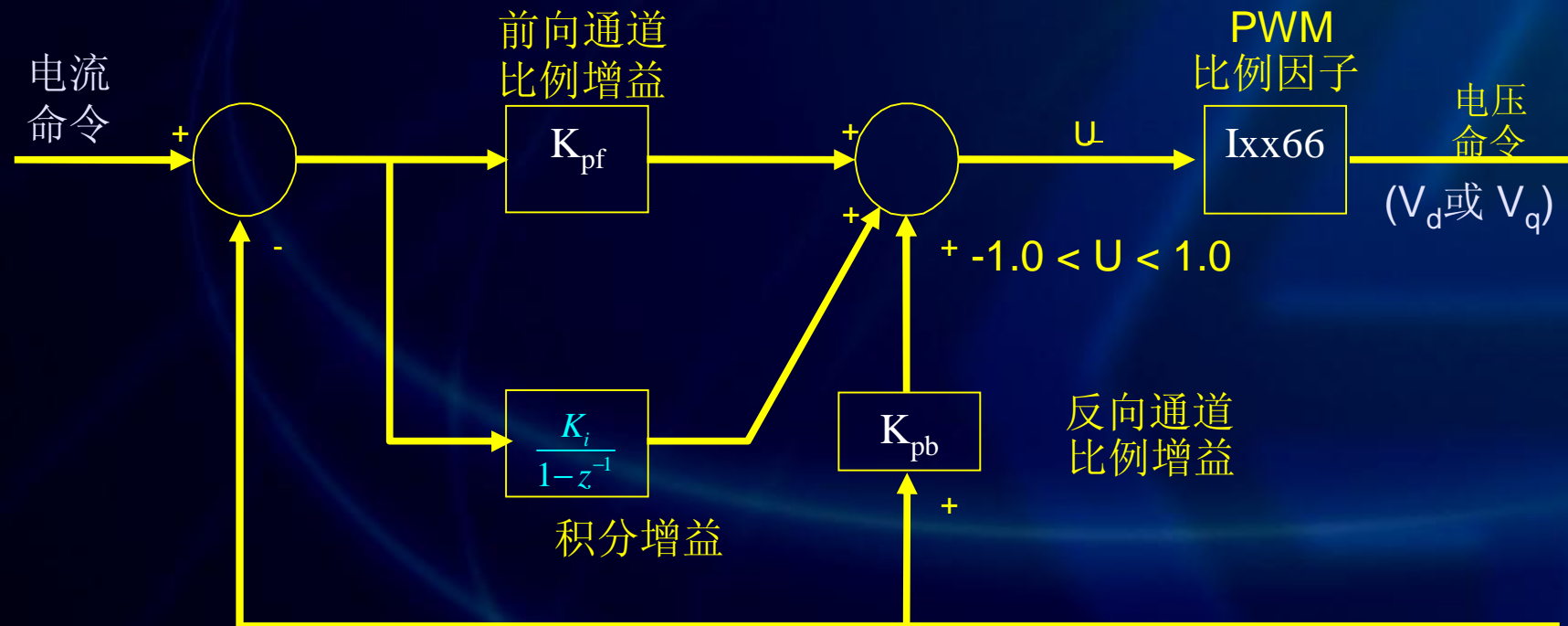
PMAC/PMAC2的模拟电流环换相



PMAC2的数字电流环换相



PMAC2的数字电流环

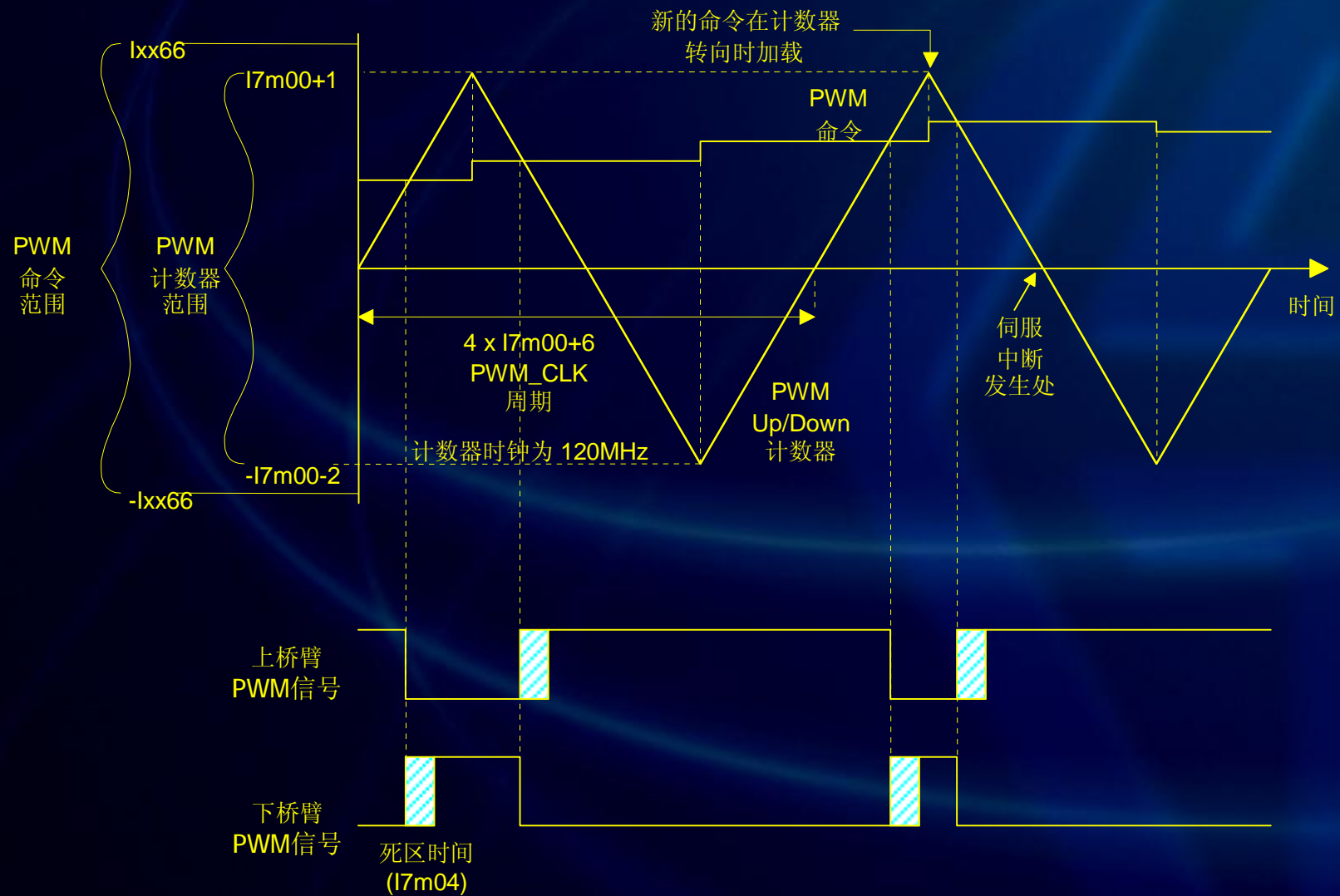


$$K_{pb} = I_{xx76} \cdot 4 \quad (0.0 < I_{xx76} < 1.0)$$

$$K_{pf} = I_{xx62} \cdot 4 \quad (0.0 < I_{xx62} < 1.0)$$

$$K_i = I_{xx61} \cdot 8 \quad (0.0 < I_{xx61} < 1.0)$$

PMAC2 数字PWM产生 (每相)



为什么在PMAC中设置数字电流环？

- n 所有的计算都在同一处理器内进行，减小了延时
 - n 可能得到更大的增益
 - n 更大的刚度，带宽，加速度
- n 数字电流环闭环
 - n 最小化高频时遇到的问题
 - n 允许更大的最大速度
- n 驱动中无控制功能
 - n 减少驱动中认为的因素
 - n 方便整定
 - n 驱动的选择只需根据功率水平

电机xx的换相变量

- n `lxx70`: 换相周期数
- n `lxx71`: 每一`lxx70`换相周期的编码计数

$$\text{编码计数/电周期} = \text{lxx71/lxx70}$$

例子: 1个极对, 4096 计数/机械转 (cts/mech rev)
 $\Rightarrow \text{lxx71/lxx70} = 4096/1$

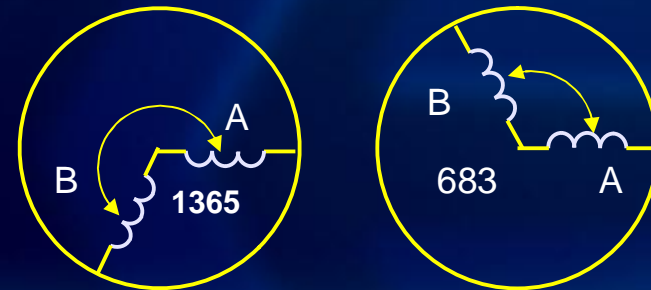
2个极对, 4096计数/机械转 (cts/mech rev)
 $\Rightarrow \text{lxx71/lxx70} = 4096/2 = 2048/1$

3个极对, 4096计数/机械转 (cts/mech rev)
 $\Rightarrow \text{lxx71/lxx70} = 4096/3$

电机x的换相变量 (续)

lxx72: 换相的相位角
相位A & B之间的角
单位: 1/2048 换相周期

n 3相: 683 或 1365



n 2 或 4 相: 512 或 1536



相位基准的I-变量

n 仅对使用增量式传感器的同步电机

n lxx73: 相位检索命令输出
搜索的幅值

n lxx74: 相位检索时间
搜索的持续时间

n 使用绝对传感器的同步电机

n lxx81: 上电换相反馈信息读取地址

n lxx91: 绝对值传感器的格式&模式

n lx75: 寻相偏移量 (单位: counts)
传感器零位和换相零位之间的差值

感抗电机的设置变量

- n Ixx77: 磁化电流
设置转子电磁场强度(&KT, KE)
感抗电机磁场所需
可在用于永磁场上
- n Ixx78: 转差率
命令转矩和转差频率之间的关系
同步电机必须为0

电机 x x的换相变量

- n lxx79: 第二相偏差(DAC 位)
- n lxx29: 第一相偏差(DAC 位)

- n lxx80: 上电模式 (是否换相)
 - 0,2= 没有使能电机, 要求使能命令
 - 1,3= 马上使能电机
 - 0,1= “两猜法” 相位检索
 - 2,3= “仿步进法”相位检索

- n lxx82: 电流环反馈地址 [仅PMAC2]
若为零, PMAC中的电流环开环

- n lxx83: 用于运行时换相信息的反馈量的地址

无刷电机的相位基准

n I. 上电相位检索

A. 两猜法 ($I_{xx80}=0,1$)

- n 随机猜测相位基准
- n 开环命令，测量加速度
- n 重复改变 90° ，
- n 从加速度计算相位基准

B. 仿步进法 ($I_{xx80}=2,3$)

- n 如同驱动带有相位偏差差项的步进电机
- n 偏差项控制相电流幅值
- n 在相位周期中两步内驱动到零位置
- n 使相位角寄存器为零

无刷电机的相位基准(续)

n II. 绝对上电参照

A. 现场运动相位搜索

- n 使用上面第二中搜索方法
- n (在相位周期零位置结束)
- n 在这个位置读绝对传感器的反馈
- n 设置 $lxx75 = -Pos/lxx70$ (偏移量项)

B. 上电时无运动参照

- n PMAC在上电时自动读绝对传感器，以确定相位基准
- n $lxx81$ 指定传感器地址
- n $lxx75$ 加到传感器位置上以得到相位位置

PMAC 相位基准的方法

方法	重复性	需要校正	备注
Hall 传感器 读	+/- 30° e	是	在第一个Hall边界或索引处校正
解析器 读	+/- 3 cts	否	步进或对称搜索建立基准
Abs 编码器 读	+/- 1 ct	否	步进或对称搜索建立基准
两步推测 检索	+/- 5-10° e	也许	要求外部轻载 微动
步进检索	+/- 1-2° e	也许	最多 1/2-周期运动
对称检索	+/- 1 ct	否	“精确相位” – 在实际应用中不可行； 只为了建立基准

换相实例操作

- n 如果演示软件中有无刷电机：
 - n 尽可能地整定好电流环
 - n 确定采用的相位检索运动类型，尝试用其它的定相方法
 - n 尝试通过“#n\$”命令完成电机的定相 (n: 电机号)
 - n 为Mxx71下载推荐的M-变量，并在电机电枢转动时在监视窗口监视它。猜想一下这一变量的显示结果？

电流限制设置 (I^2T 保护)

PMAC 电流限制参数

n lxx69: 瞬时电流限制

- n 保和限制
- n 速度模式驱动时，用作速度限制

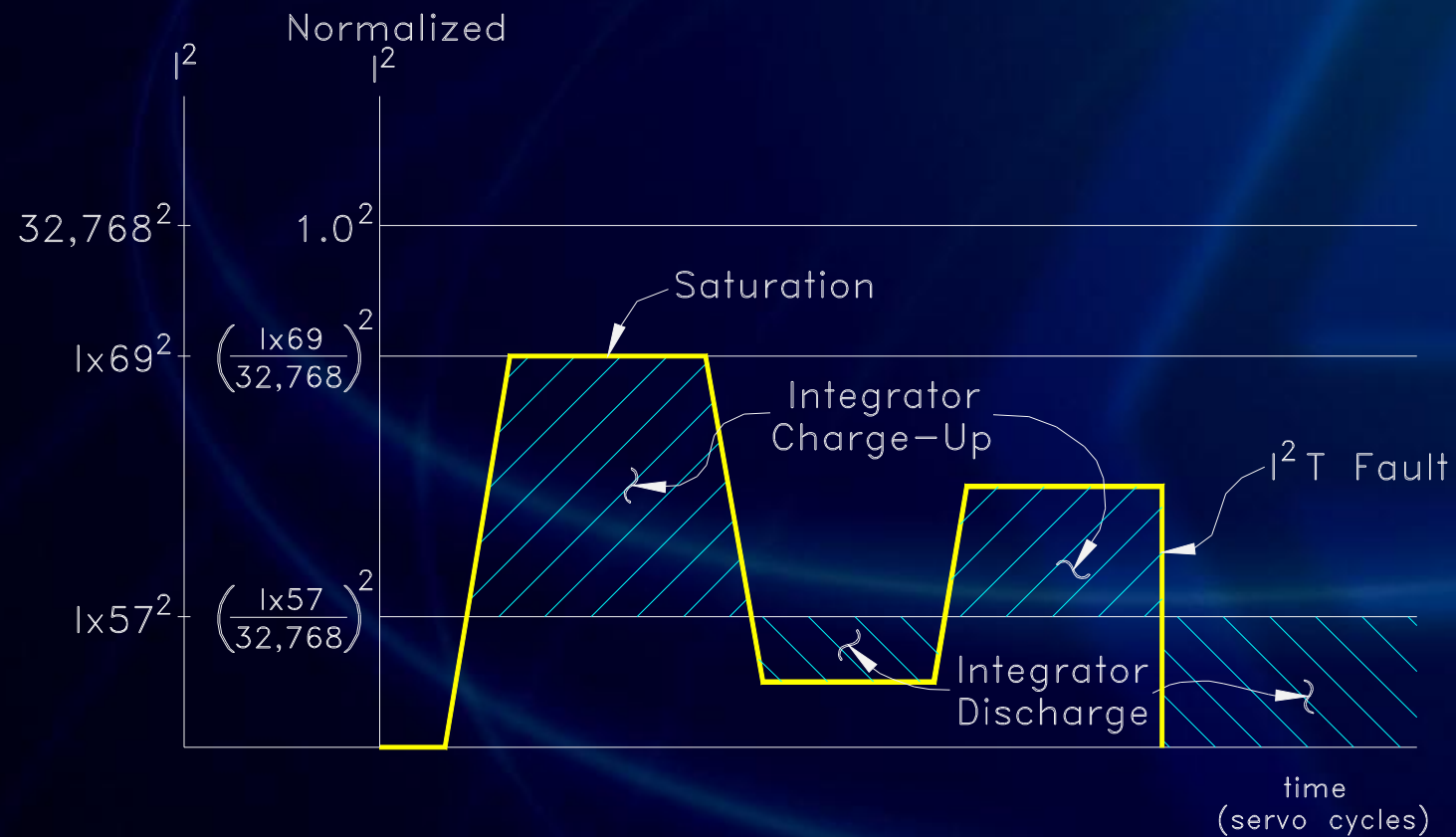
n lxx57: 连续电流限制

- n 电流值大于lx57将增加电流积分值
- n 电流值小于lx57将减小电流积分值
- n 直接PWM方式下使用测量电流值检测是否超过lxx57
- n 其它方式下使用命令电流值

n lxx58: 电流积分值限制

- n 当电流积分器超过lx58时:
 - n 如同服务器发生故障一样将停止电机
 - n 同时置“积分电流故障”位

PMAC I²T 保护特性



PMAC 补偿表

PMAC 补偿表

标准螺距补偿

例. $\Delta x = f(x)$

- n 使用旋转编码获得接近线性编码器准确程度
- n 使系统达到使用线性传感器效果
- n 输入误差到PMAC



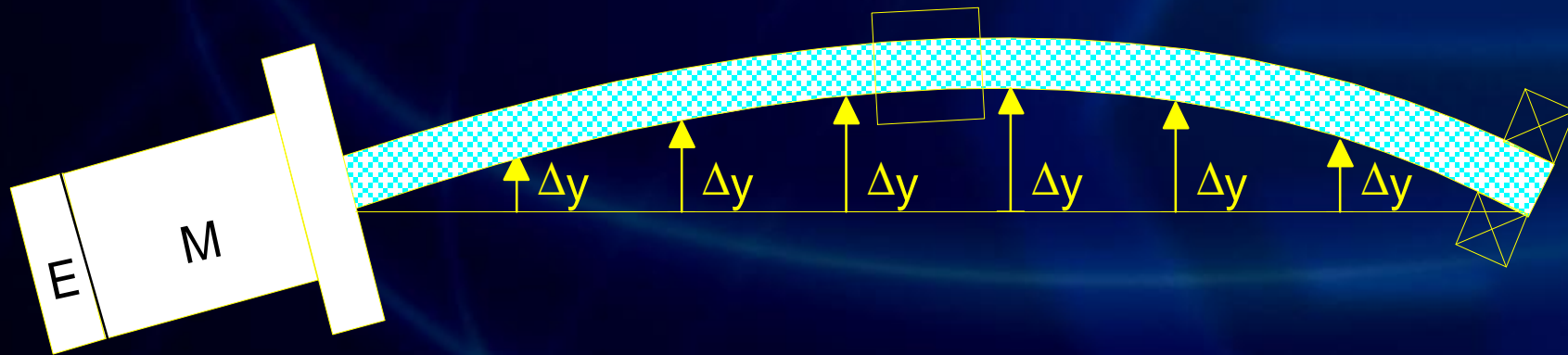
PMAC 补偿表

交叉轴补偿

例. $\Delta y = f(x)$

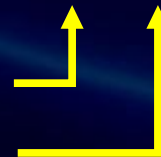
n 用于弯曲丝杆

n 用于设置电子凸轮表



DEFINE COMP 100, #1, #2, 100000

表长
源电机

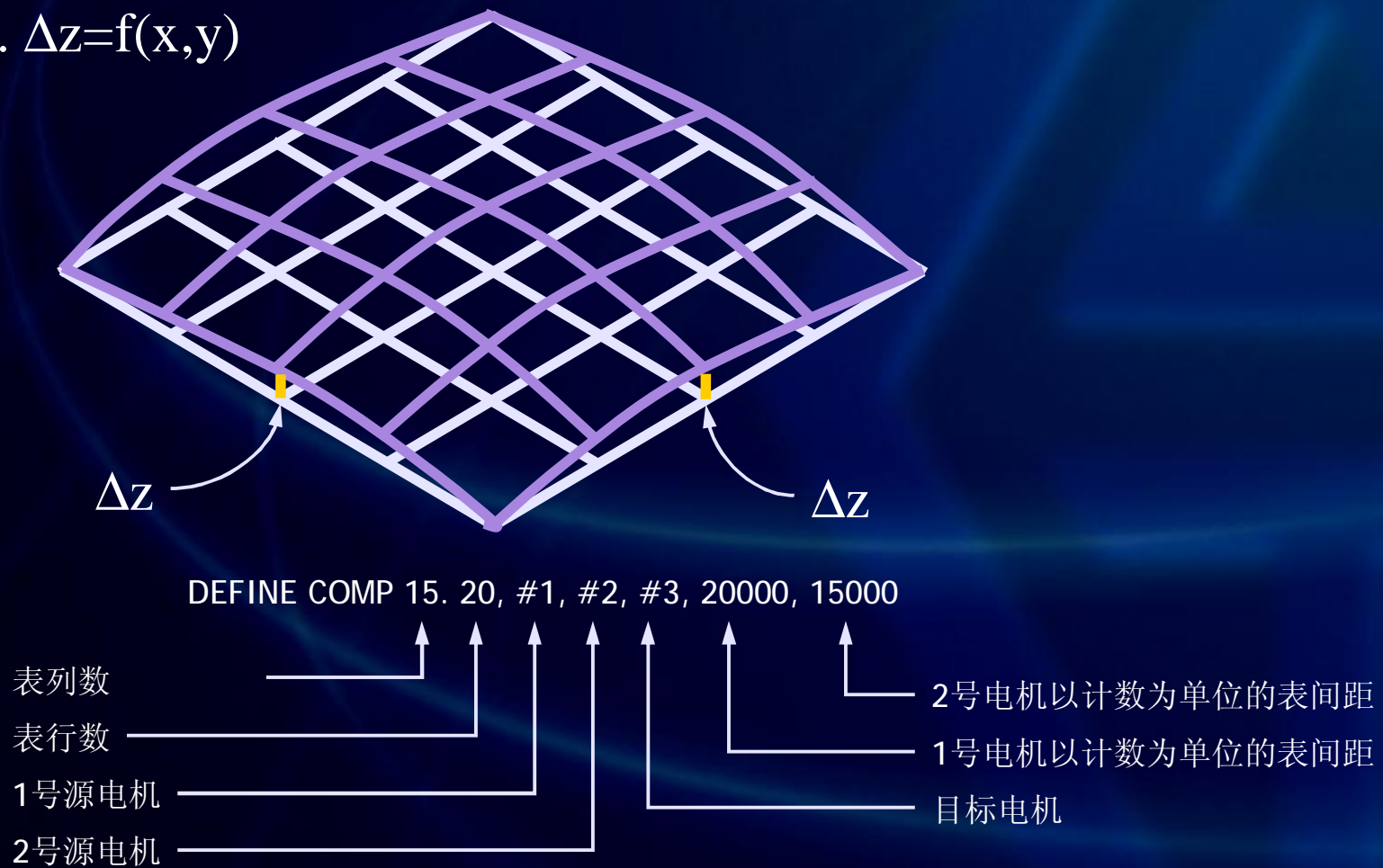


以计数为单位的表间距
目标电机

PMAC 补偿表

2D (平面) 补偿表

例. $\Delta z = f(x, y)$



PMAC 补偿表 用于激光打标系统

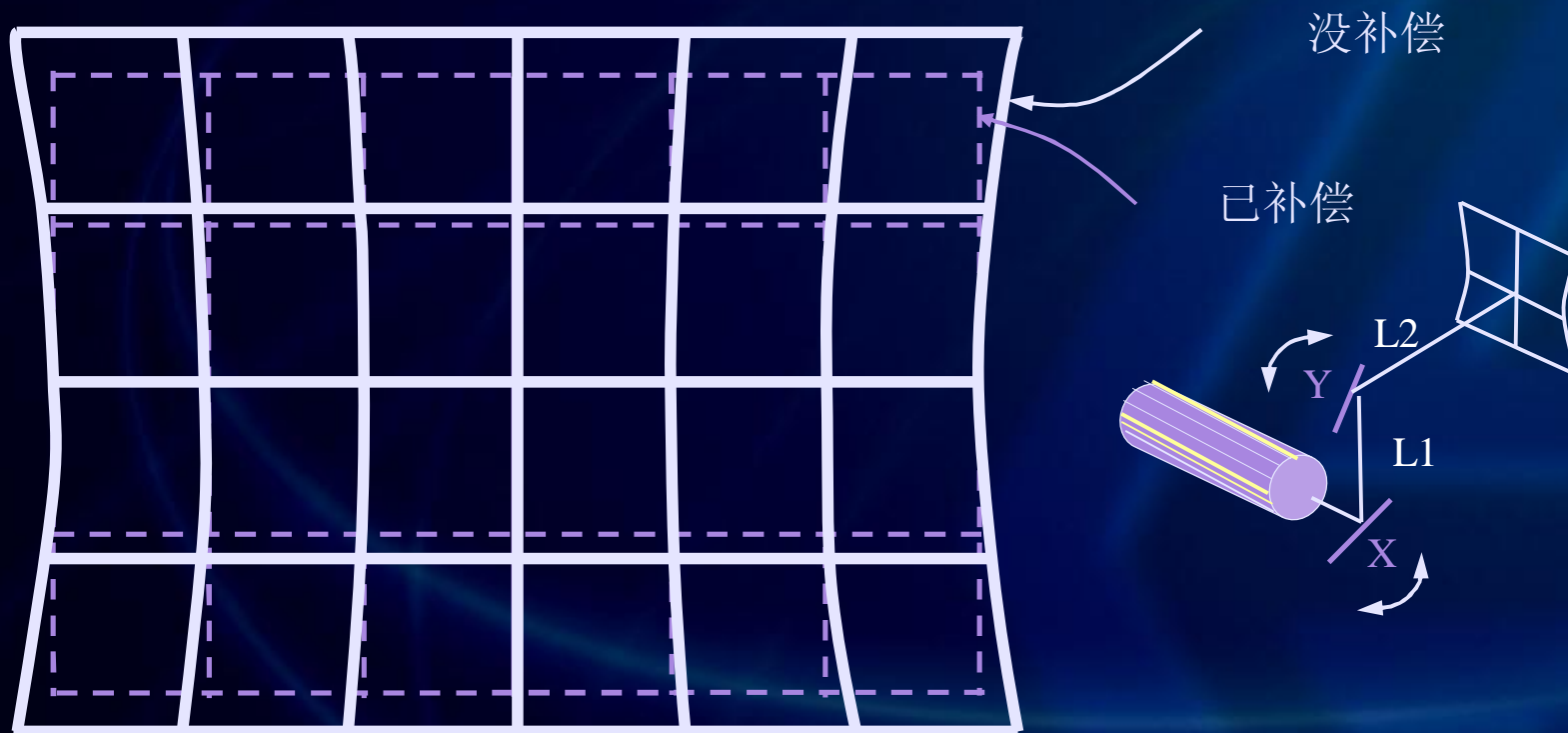


表1: $Dy=f(y)=\arctan(y/L2)-(y/L2)$

表2: $Dx=f(x,y)=\arctan[x/(L1+\sqrt{L2^2-y^2})]-[x/(L1+L2)]$

PMAC 的电机齿隙参数

- n lxx86:** 齿隙大小 (1/16计数)
反向时额外的 {隐藏} 位置改变
- n lxx85:** 齿隙拉紧率 (1/16 计数/后台周期)
反向时齿隙改变率
为保持平滑的过渡此值应尽可能高
- n lxx87:** 齿隙滞环 (1/16 计数) (全局的)
添加齿隙补偿时需要的反向大小阈值
设置为非零，防止抖动

PMAC的电机齿隙表

n #x DEFINE BLCOMP {#entries}, {count span}

n 位置相关的齿隙

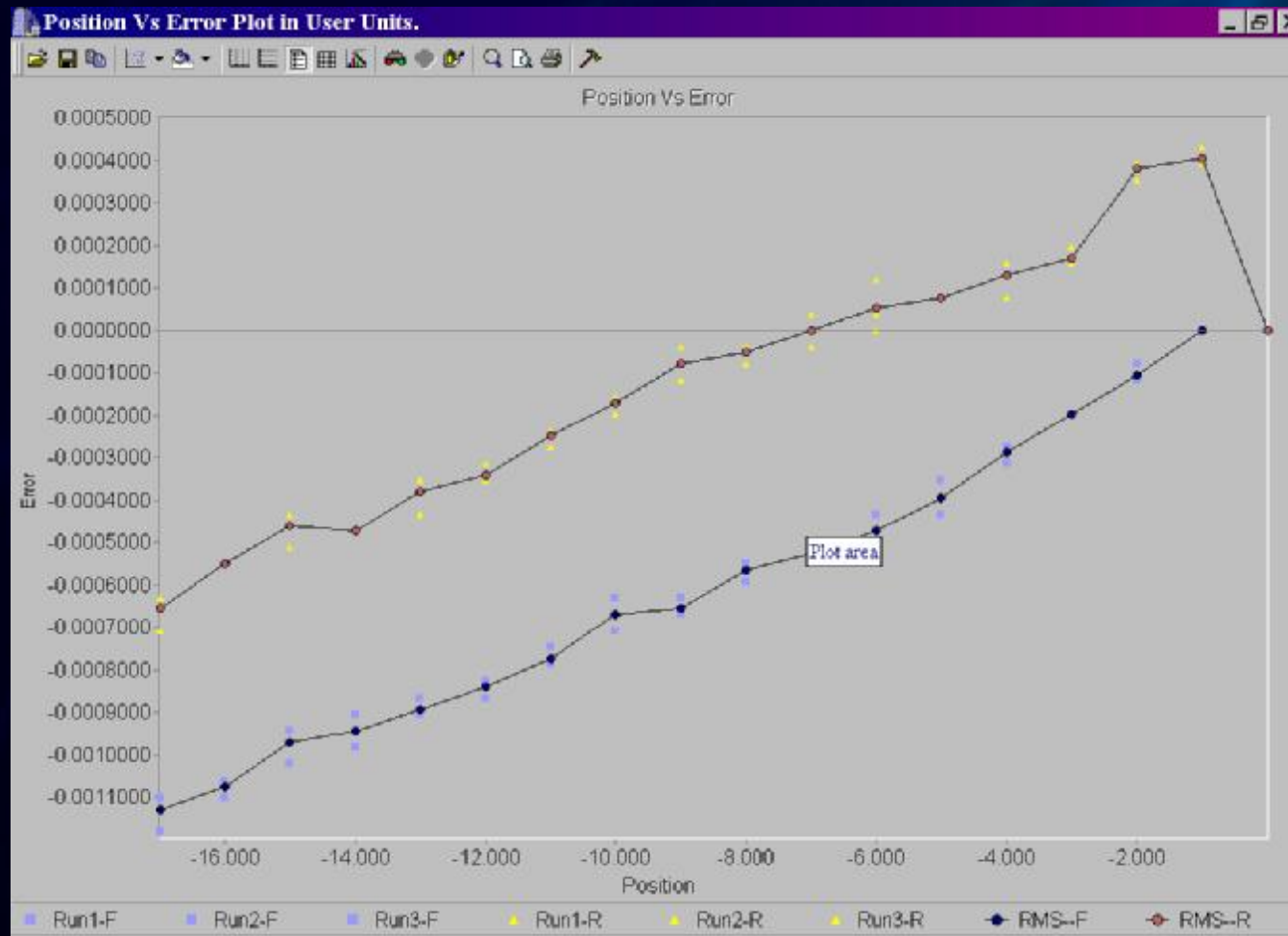
n 加入 **lx86** {常数} 齿隙

n 零号电机的齿隙表为零

n 与补偿表一起使用来添加双向补偿

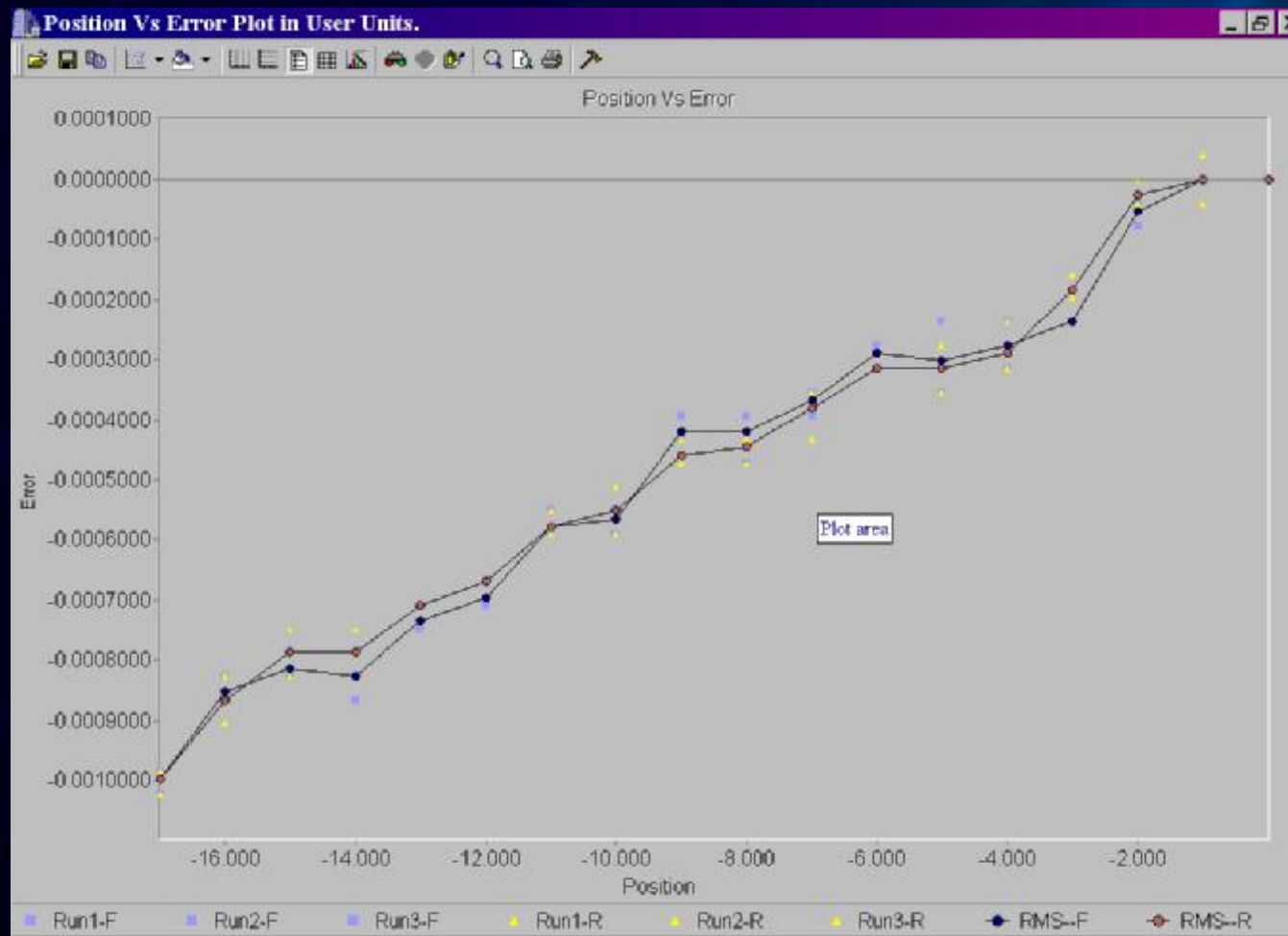
n 保留正向和反向补偿之间的差异

PMAC的补偿结果



没有补偿的运动结果

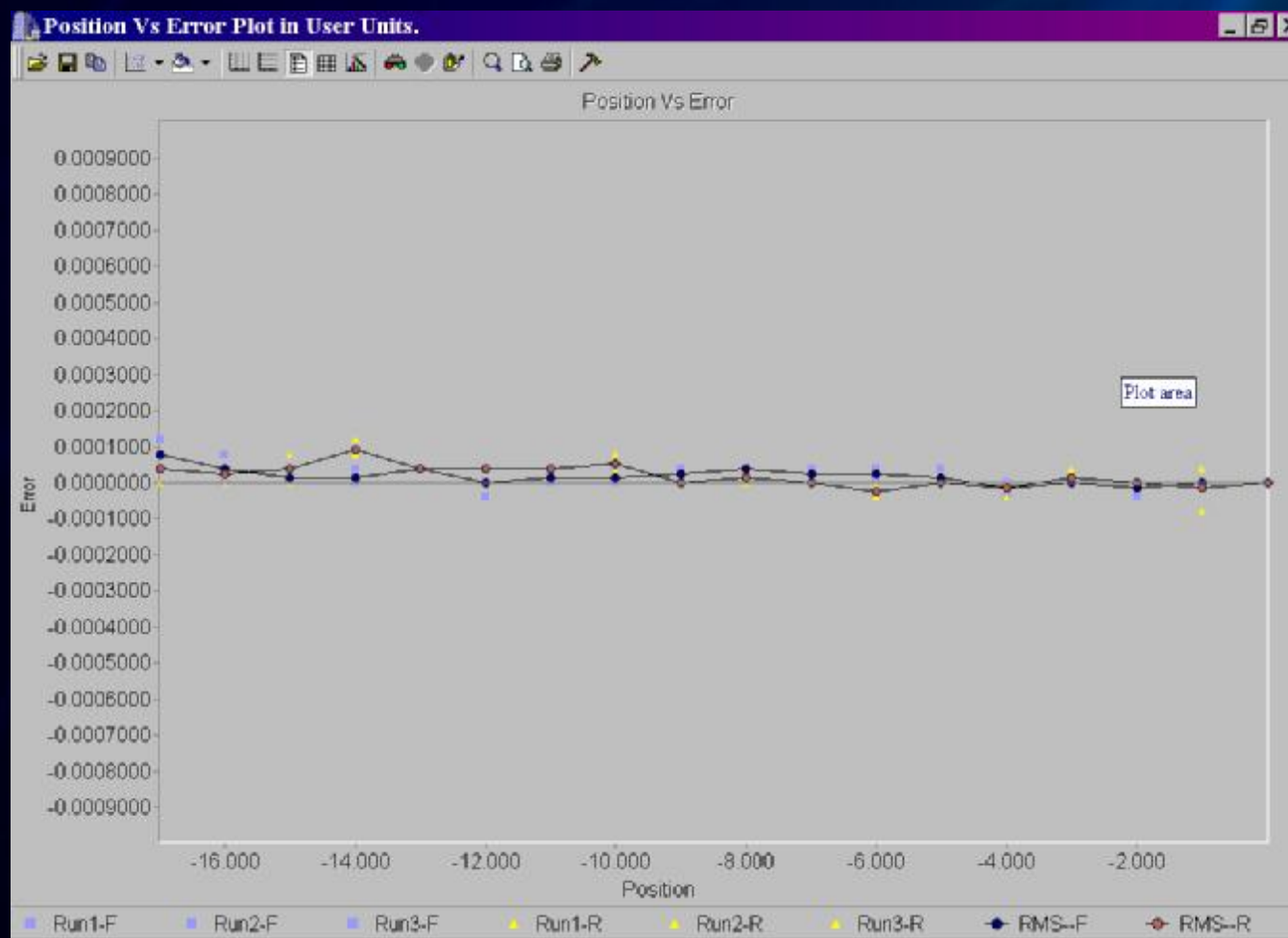
PMAC的补偿结果(续)



齿隙补偿表激活

螺距补偿表没激活

PMAC的补偿结果(续)



齿隙和螺距补偿表激活

PMAC 电机转矩补偿表

n #x DEFINE TCOMP {#entries}, {count span}

n 调节伺服输出使之成为位置的函数

n 电机脉动转矩补偿

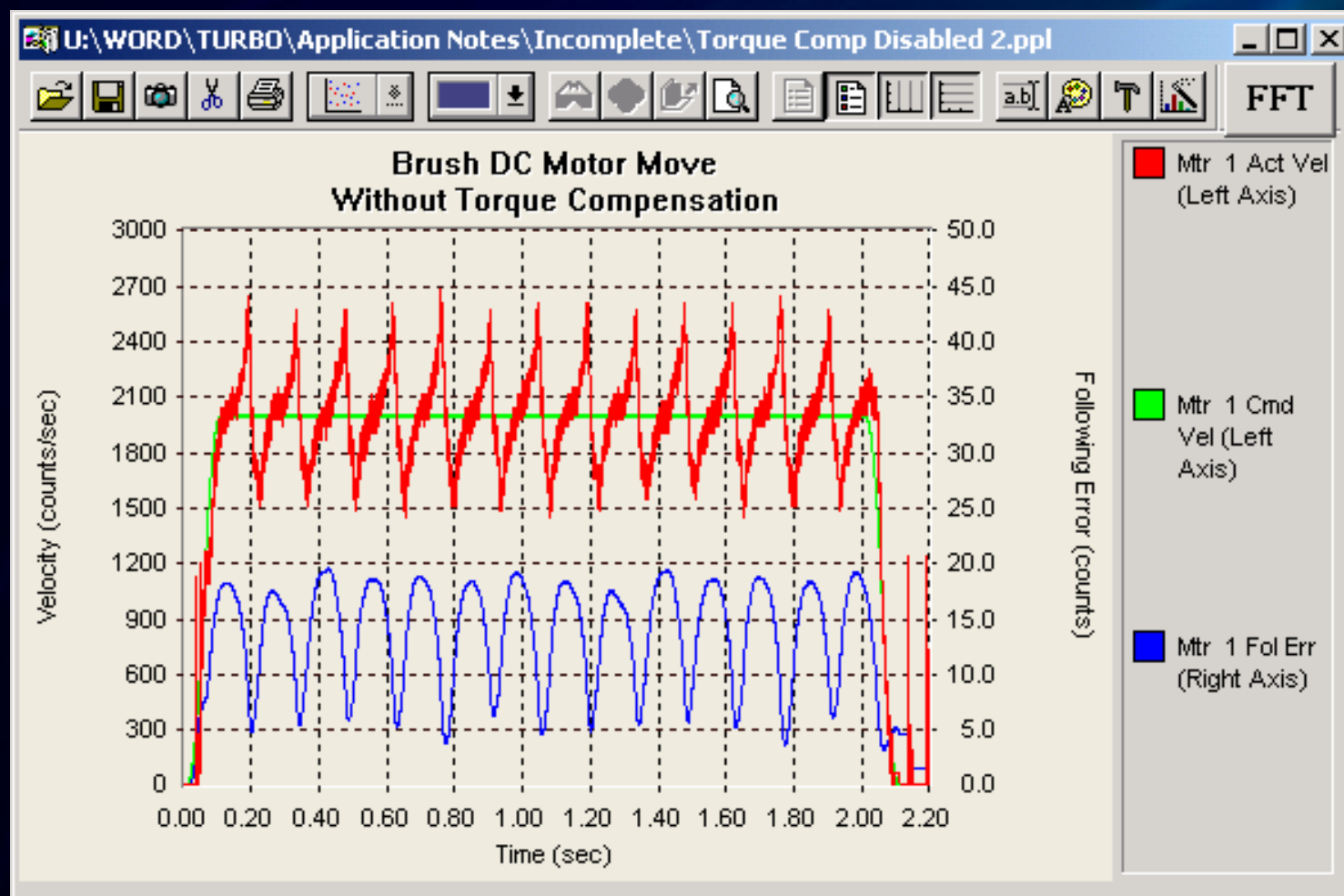
n 自动翻转 — 每个机械周期

n 如何设置：发命令让电机转到表中期望的每一点

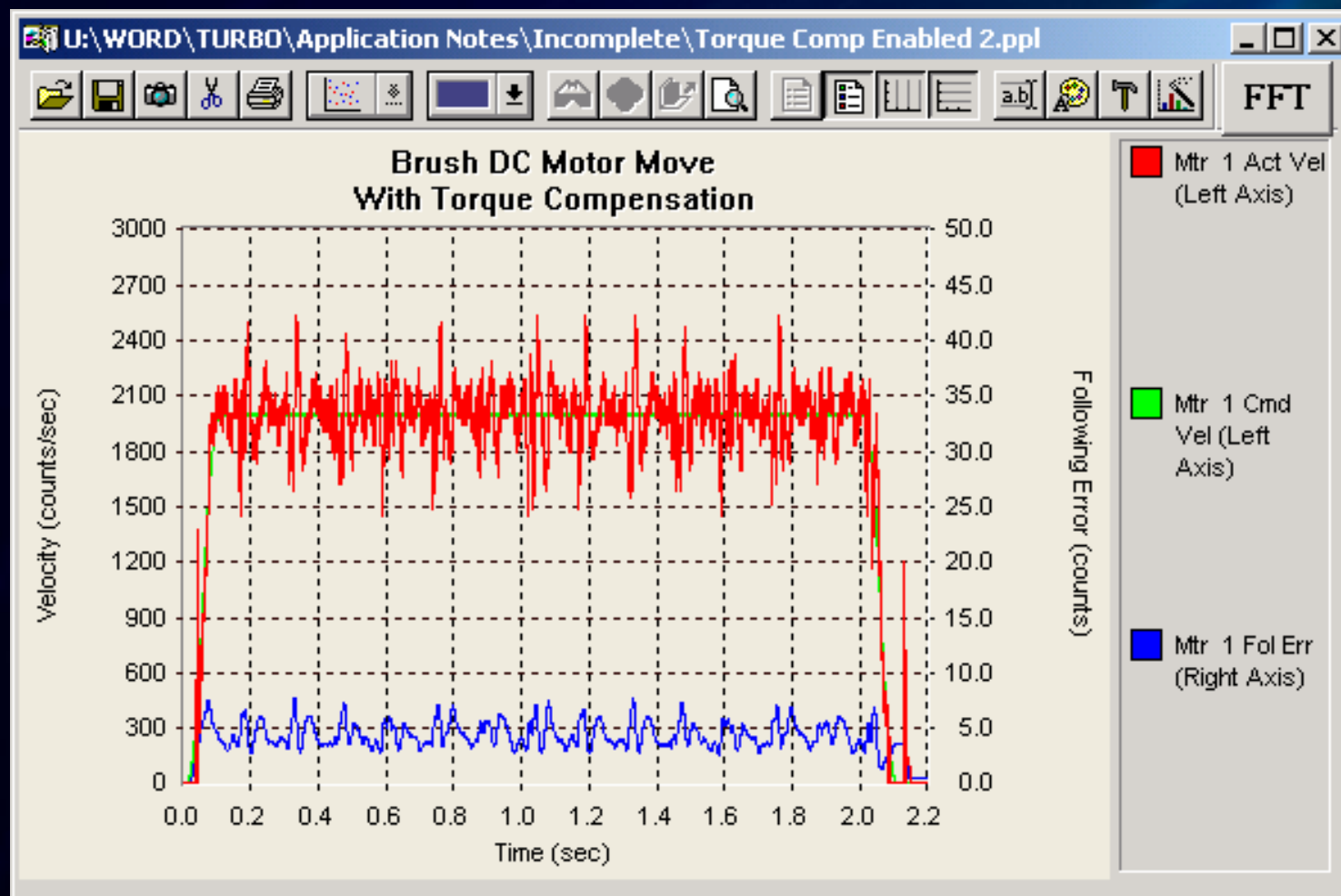
n 激活伺服环中积分增益使位置误差为零

n 运动到指定位置并稳定下来后，读伺服输出

转矩补偿前

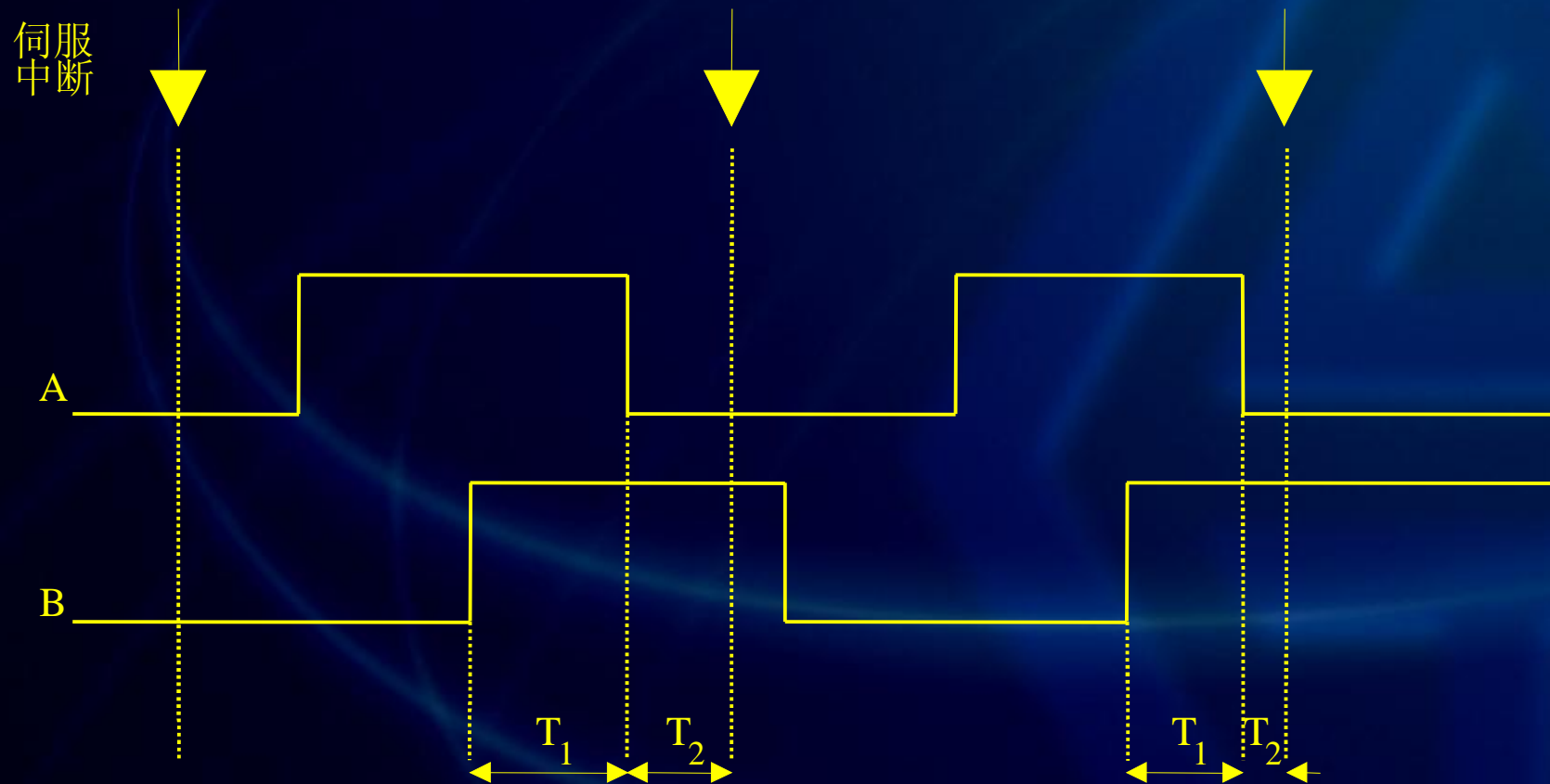


转矩补偿后



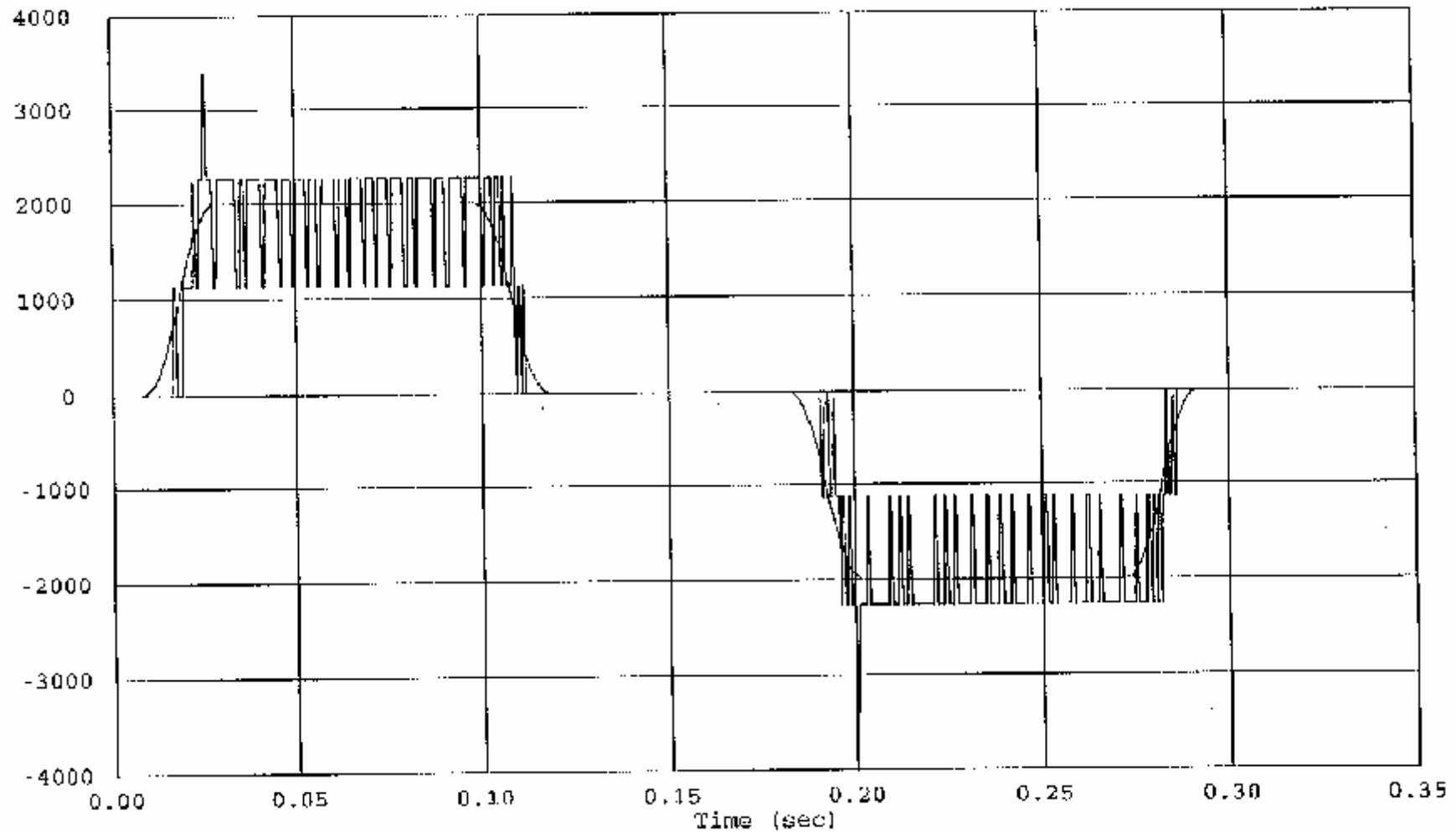
PMAC 1/T 扩展

PMAC 1/T 扩展



$$V_n = \frac{K}{T_1} \quad P_n = \text{计数器} \pm \frac{T_2}{T_1}$$

Non 1/T Velocity with Non 1/T Interpolation

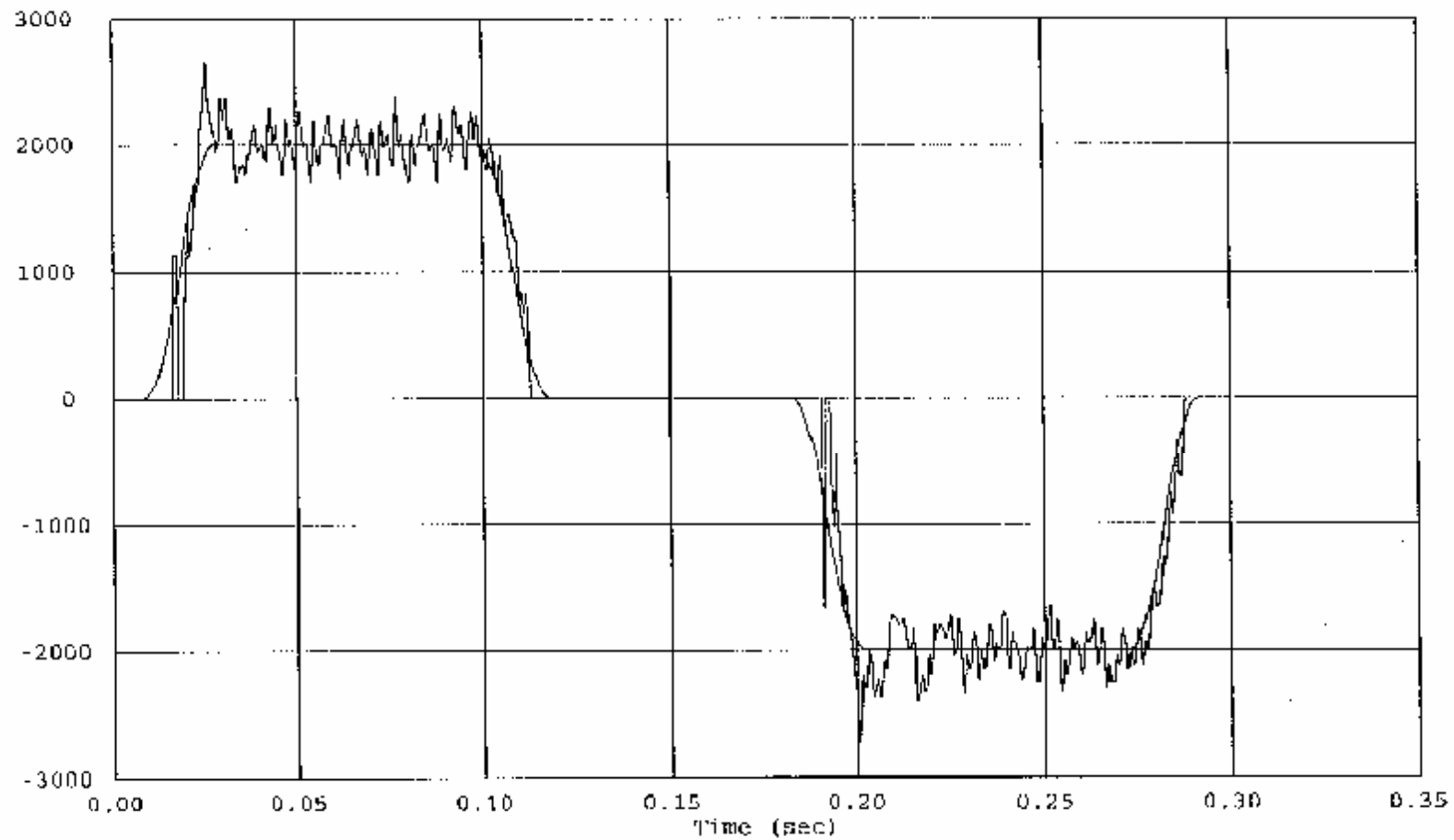


Mtr 1 Act Vel (cm/sec)

Mtr 1 Cmd Vel

MAC Executive for Windows
Release 1.50, 07/03/1996 - 11:19
Copyright © 1994,95,96, Della Tau Data Systems, Inc.
Development:1

1/T Velocity with Non 1/T Interpolation

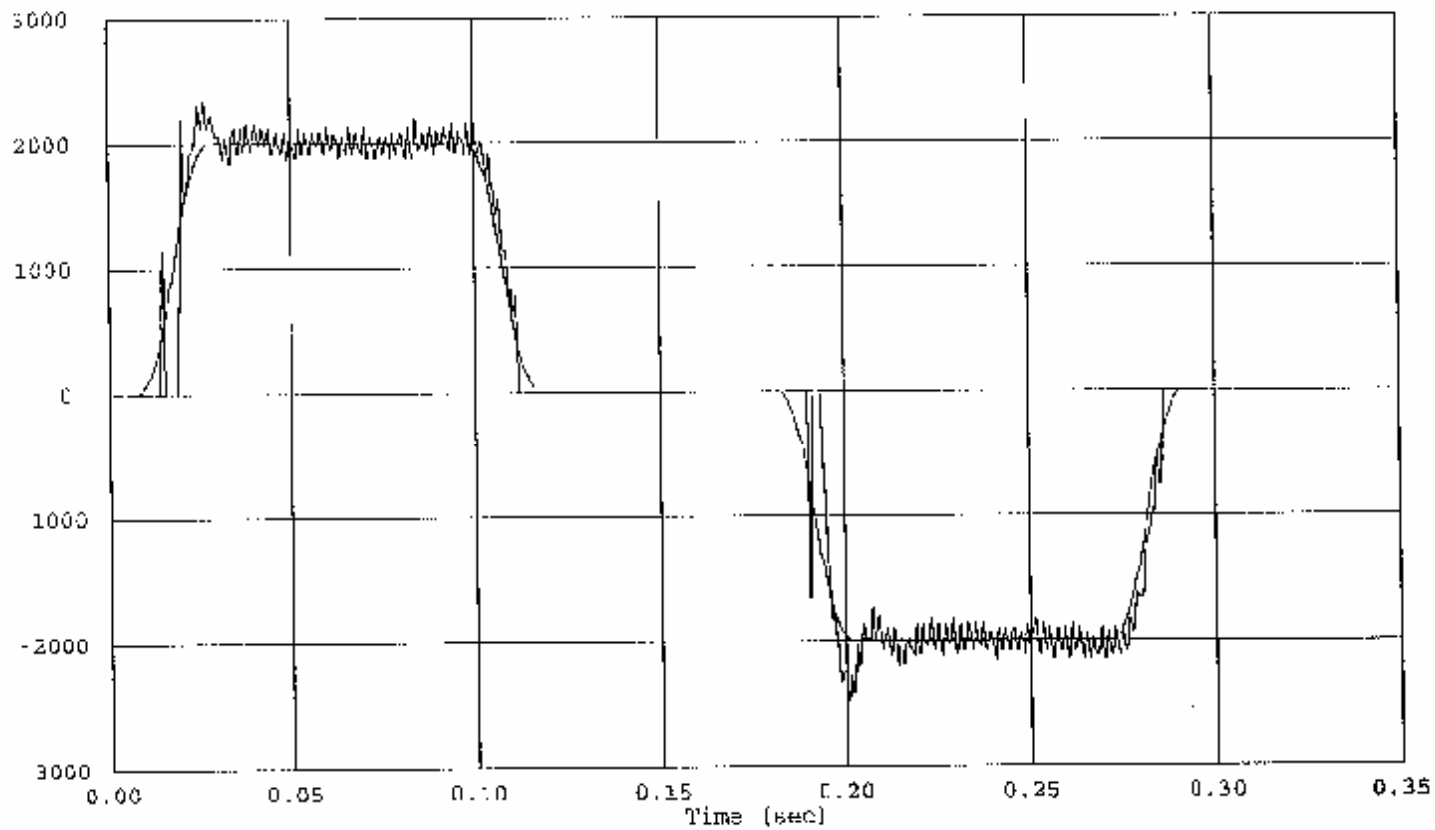


Min 5 Acs Vel (ch/sec)

Min 5 Cmd Vel

PMAC Executive for Windows
Release 1.50 07/03/1995 - 11:03
Copyright © 1994,95,96, Delta Tzu Data Systems, Inc.
Development:1

1/1 Velocity with 1/1 interpolation

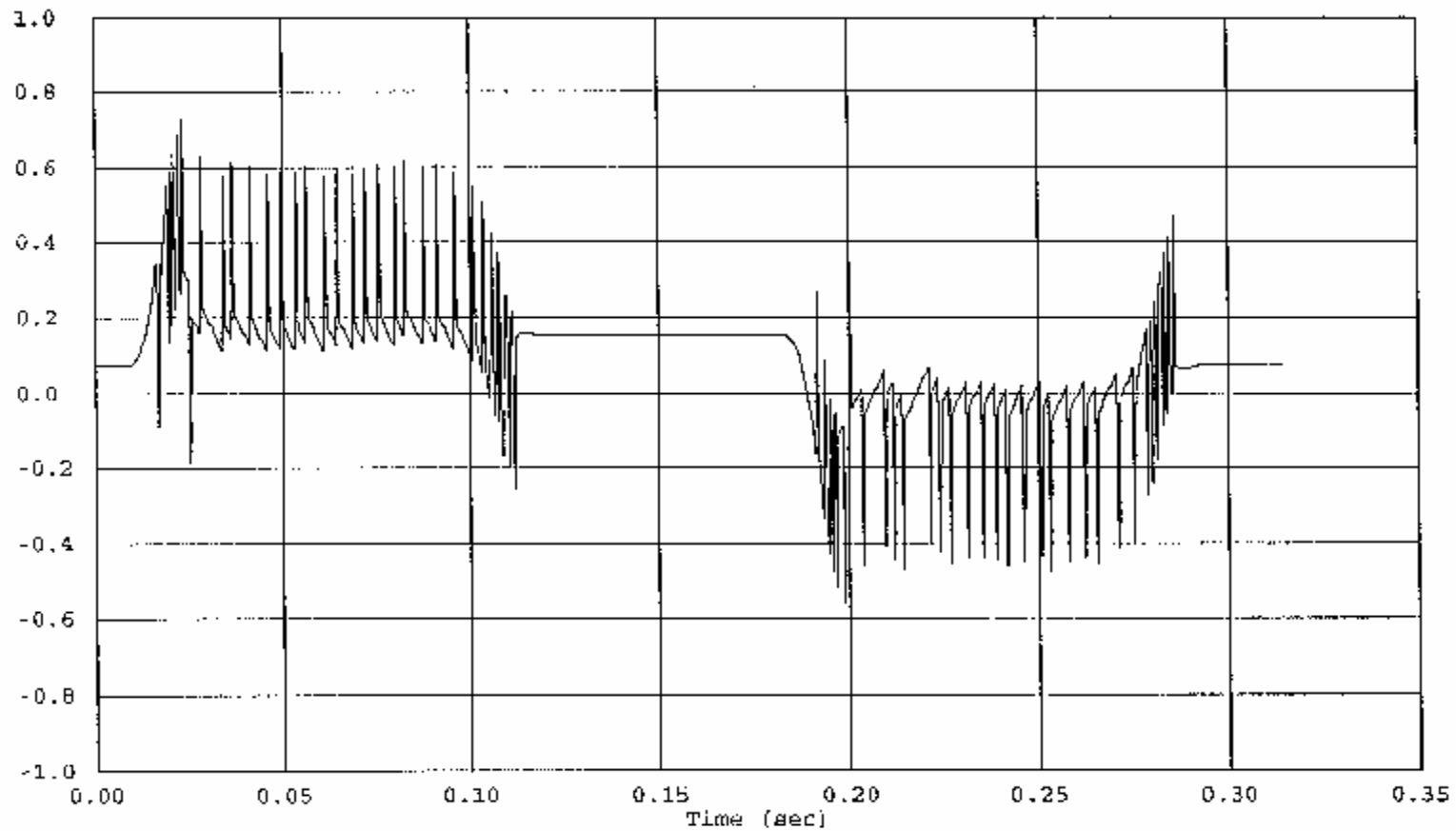


Mir 5 Act Vel (cm/sec)

Mir 5 2nd Vel

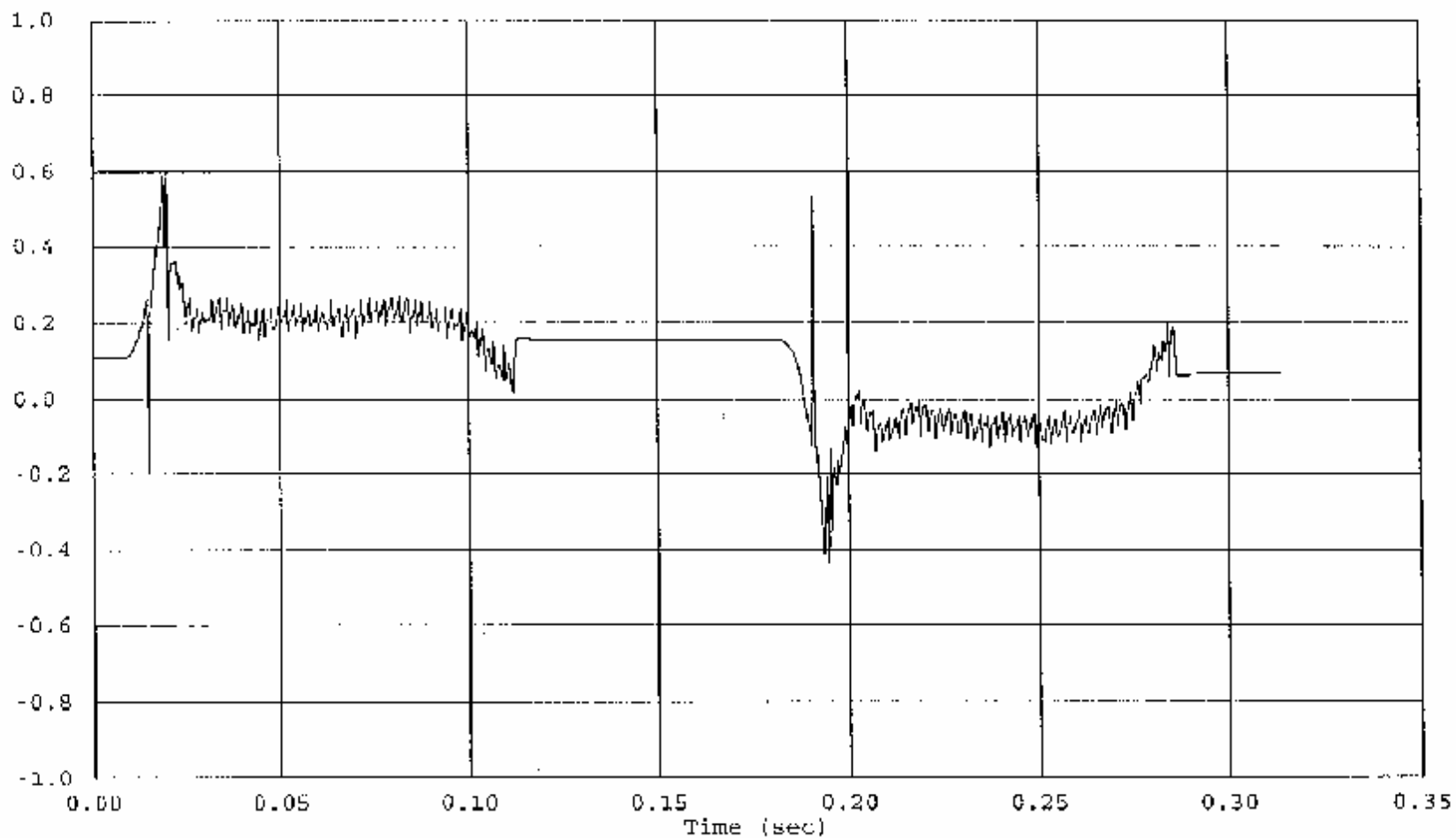
PMAC Executive for Windows
Release 1.50, 07/03/1998 - 11:03
Copyright © 1994,95,96, Delta Tau Data Systems, Inc.
Development:1

DAC Output with Non 1/T Interpolation



PMAC Executive for Windows
Release 1.5D, 07/03/1996 - 11:17
Copyright © 1994,95,96, Delta Tau Data Systems, Inc.
Development:1

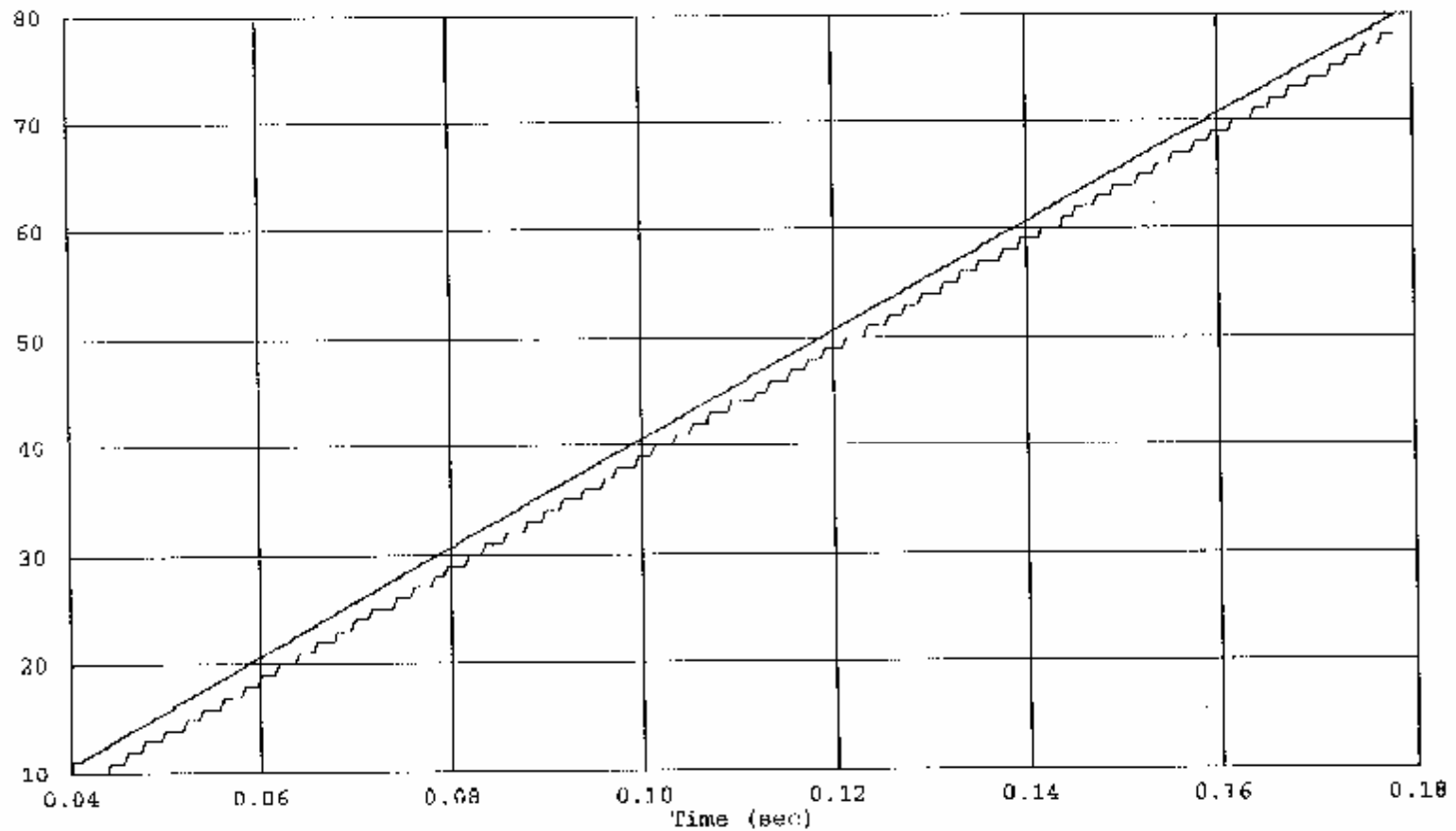
DAC Output with 1/T Interpolation



Plot 1: DAC Output (volts)

PMAC Executive for Windows
Release 1.50, 07/03/1998 - 11:02
Copyright © 1984,95,96, Delta Tau Data Systems, Inc.
Development:1

Non 1/T Position with Non 1/T Interpolation

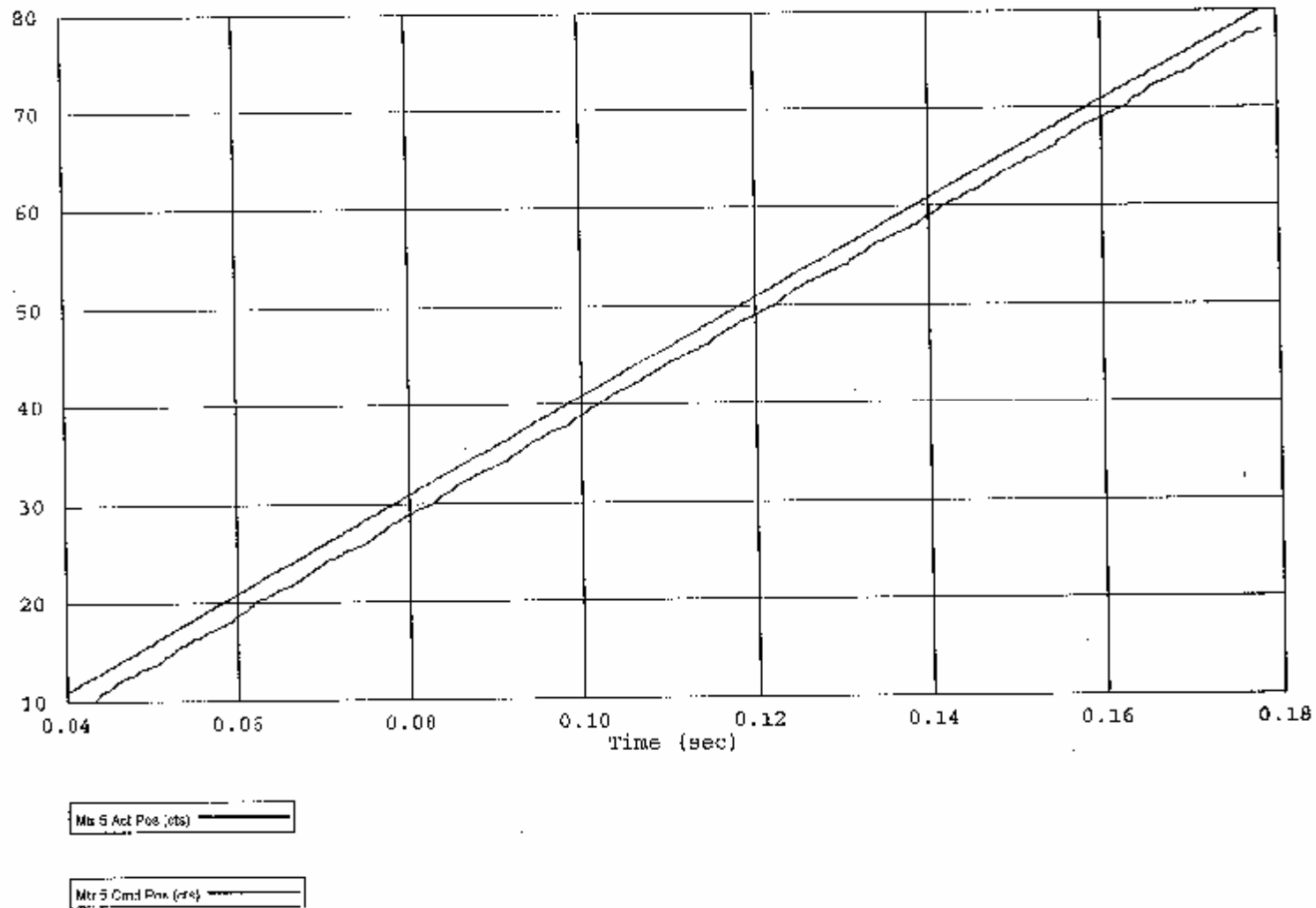


Mtr 1 Act Pos (cts)

Mtr 1 Cmd Pos (cts)

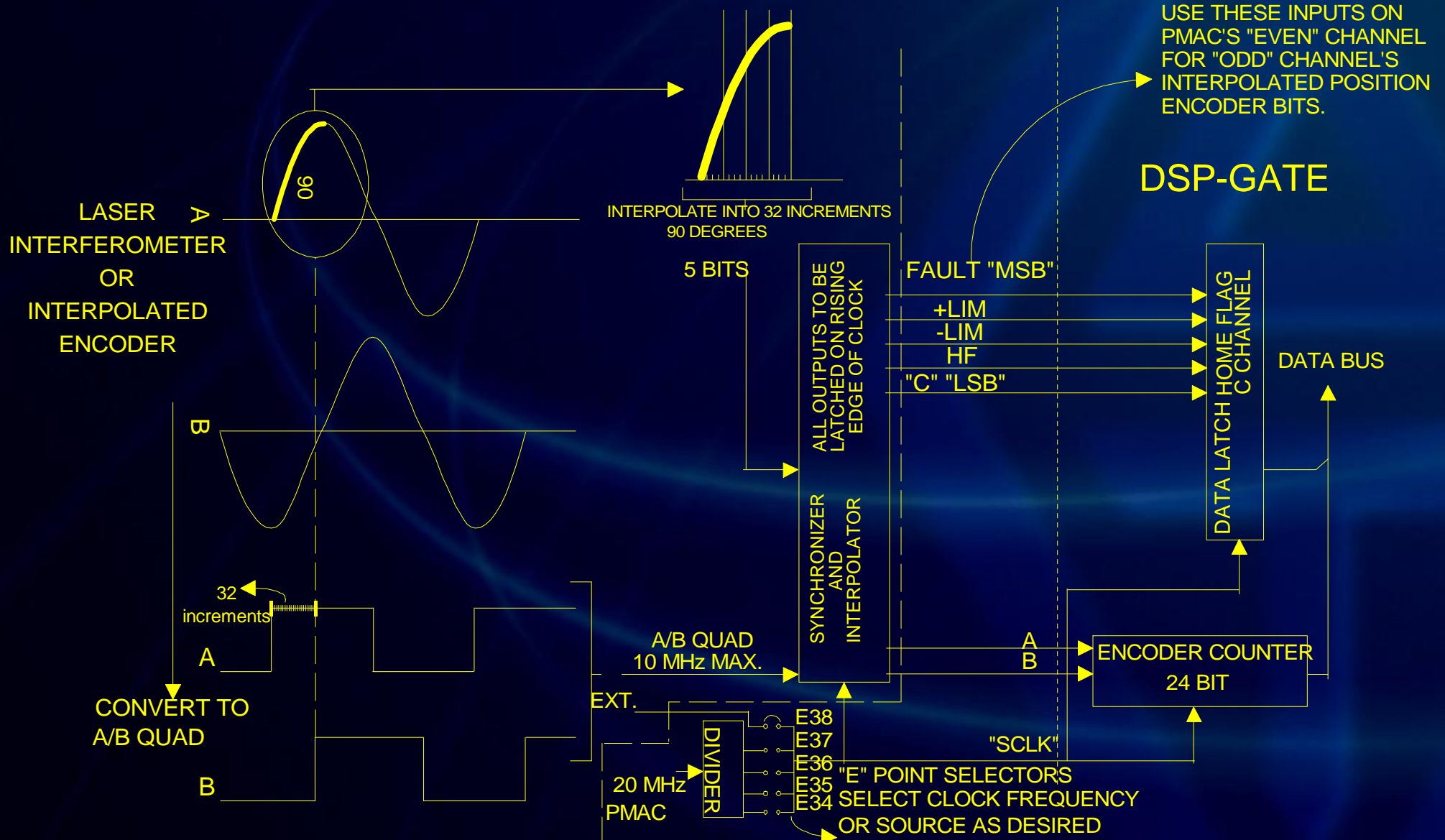
PMAC Executive for Windows
Release 1.50, 07/03/1996 - 11:02
Copyright © 1984,95,96, Delta Tau Data Systems, Inc.
Development L1

1/T Position with 1/T Interpolation



编码器反馈差值

NOTE: A 4 AXIS PMAC CAN HANDLE ONLY 2 AXIS OF CONTROL WHEN USING BIT INTERPOLATION FOR ENCODER.



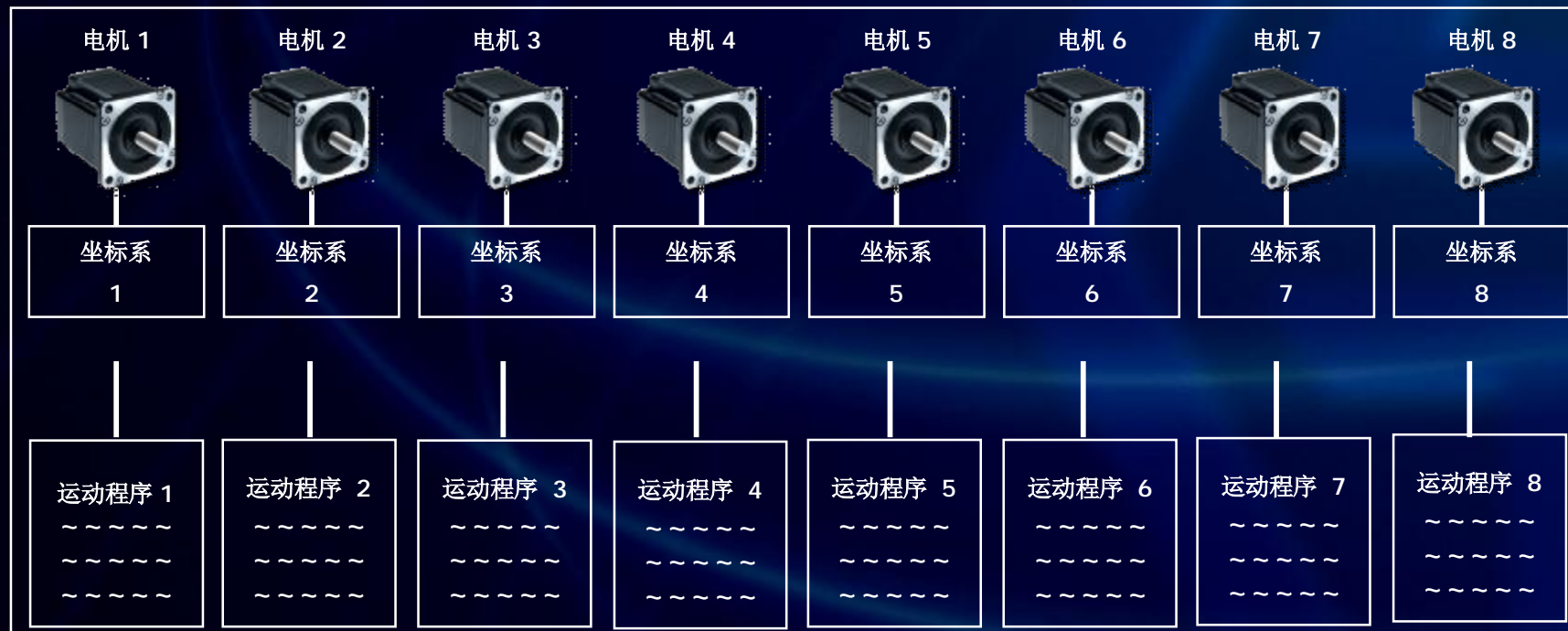
PMAC 坐标系统

什么是坐标系？

- n PMAC中的坐标系指的是为了各轴间同步运动目的而组织起来的电机组。(等价于数控系统中的通道)
- n 根据程序的期望动作，电机被分组到坐标系中
- n 一个坐标系 (即使只有一个电机)可以运行运动程序；而一个电机不行
- n non-Turbo PMAC有8个坐标系 (&1...&8)，Turbo PMAC有16个坐标系

坐标系统

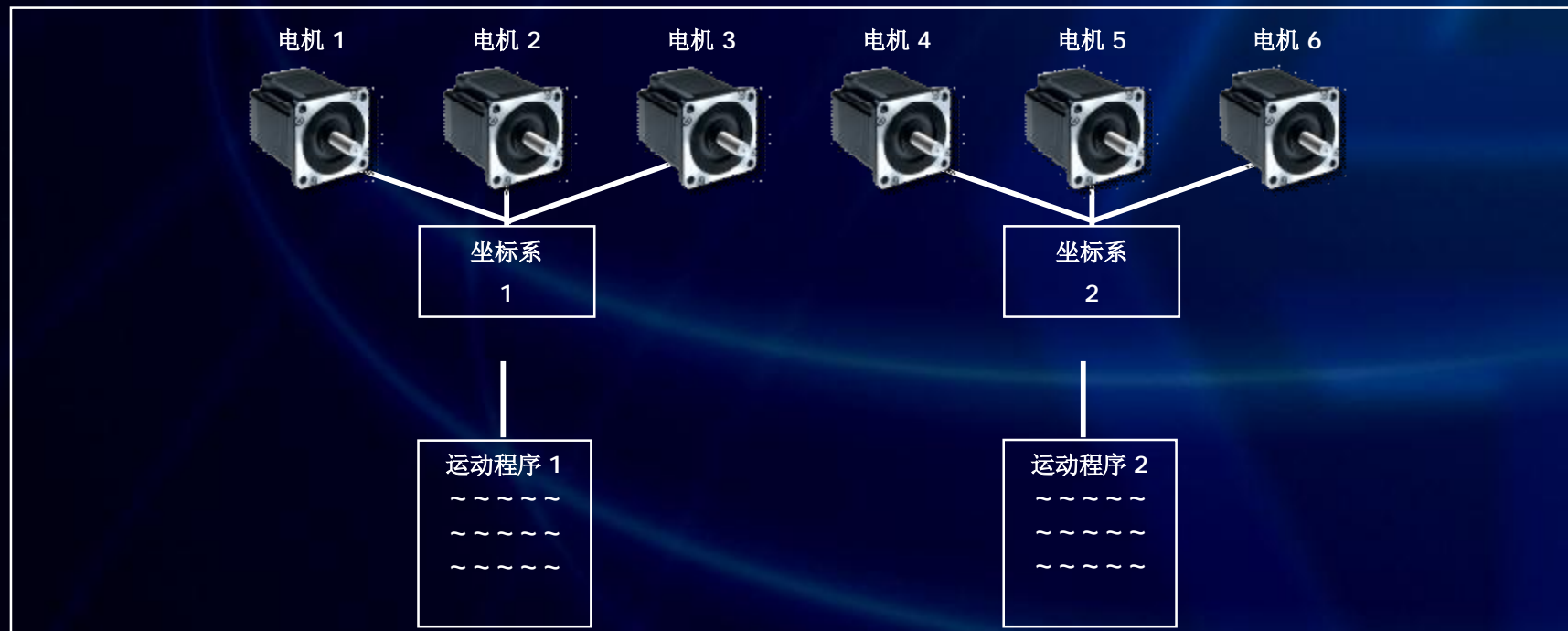
- n 将电机**分到**不同坐标系下的目的：
 - n 一个机器的运动完全独立于其它机器。



坐标系统

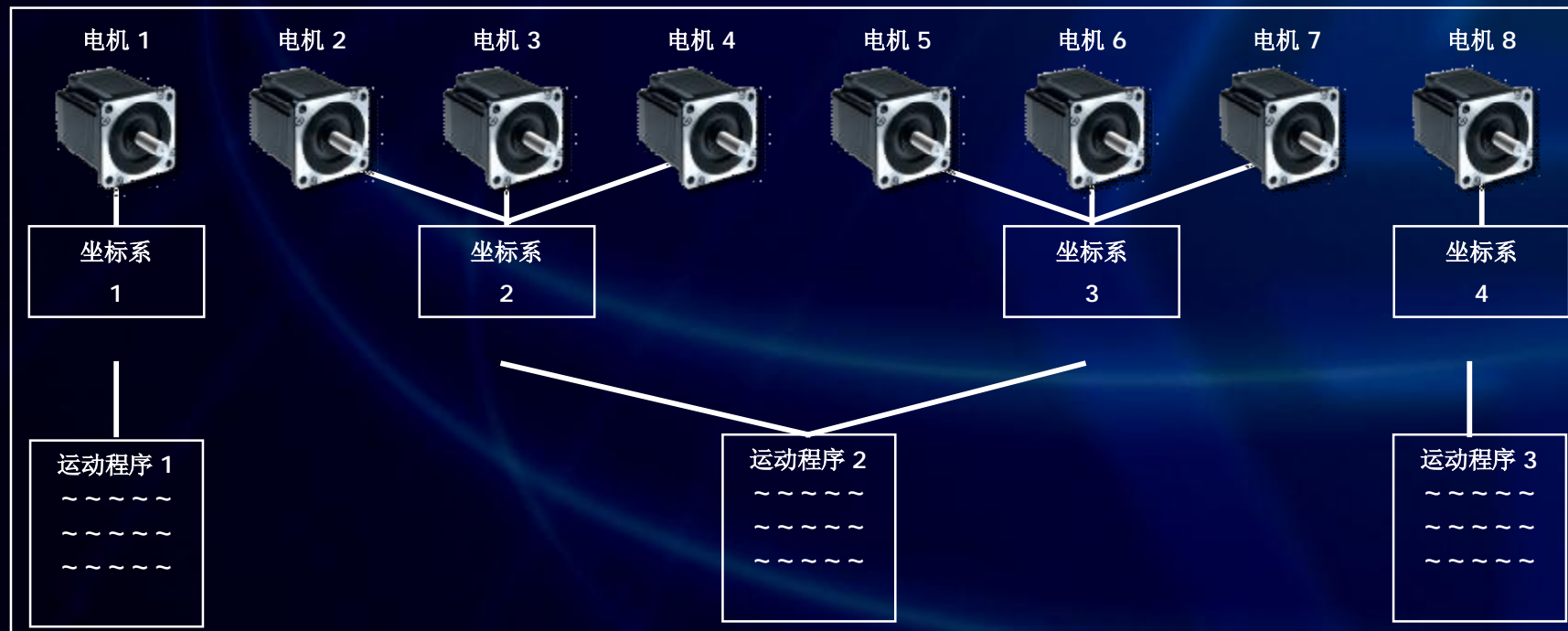
n 将电机**分组到**不同坐标系下的目的:

n 将电机分组可使同一坐标系下的电机运行运动程序时准确的进行同步运动。



坐标系统

- n 坐标系可以在不同的或者重叠的时间内，运行不同的或者相同的程序。



什么是轴？

- n 轴是坐标系的一个元素。通过分配给电机进行定义，同时有比例因子和偏移量两个参数。
- n 允许的轴的取名： [X,Y,Z] [U,V,W] [A,B,C]
- n 一对一的匹配
 - n 单个电机分配给单个轴 #1->2000x+500
- n 多电机轴
 - n 将多个电机分配给同一坐标轴 龙门系统;
#1->2000X
#2->2000X
- n 虚拟轴
 - n 坐标系中的轴可以不与实际电机相关(一个“虚拟”轴)

轴的类型和属性

- n 笛卡尔轴: **[X,Y,Z,]** **[U,V,W 辅助]**
 - n 将两个或三个轴分组, 使得沿着轴上的运动为两个或三个电机的线性组合运动
- n 笛卡尔轴: **[X,Y,Z]**
 - n 刀具半径补偿
 - n 圆弧插补
 - n 轴转换矩阵
- n 旋转轴: **[A,B,C]**
 - n 允许转动 [Ix27]
- n 进给轴: **[X,Y,Z]** 缺省

定义坐标系

- n 坐标系的定义：通过寻址该坐标系并将电机分配其所属的轴。
- n 运行前，给每个电机n使用 **undefine**, **undefine all**语句，或者 **#n -> 0**，以清除所有存在的定义。

Undefine all

&1

#1->X

#2->Y

&2

#3->Z

#4->Y

轴定义语句

- n 通过轴定义语句来建立坐标系。
- n 通过将一个电机(数字编号)与一个或多个轴(字母编号) 匹配来完成轴定义。
(例: #1-> 10X + 20Y)
- n 电机映射到轴字母, 并带有可选的比例因子和从原点的偏移距离。
- n 比例因子表示每一个用户单位对应多少个编码器计数。

例如: **#1 -> 10000X + 5000**

轴的定义语句(续)

&1

#1->X

&2

#2->X

{ 该定义是允许的。
两个电机在不同的坐标系下定义
为X轴。

&1

#1->X

#2->X

{ 该定义是允许的。
电机将在同一X轴轨迹上运动,
如龙门系统。

轴定义语句

&1

#1->X

#1->Y



该定义不允许。

一个电机不能在程序的同一时间中执行不同的运动轨迹。

第一个轴定义将被第二个轴定义替代。

&1

#1->X

&2

#1->X



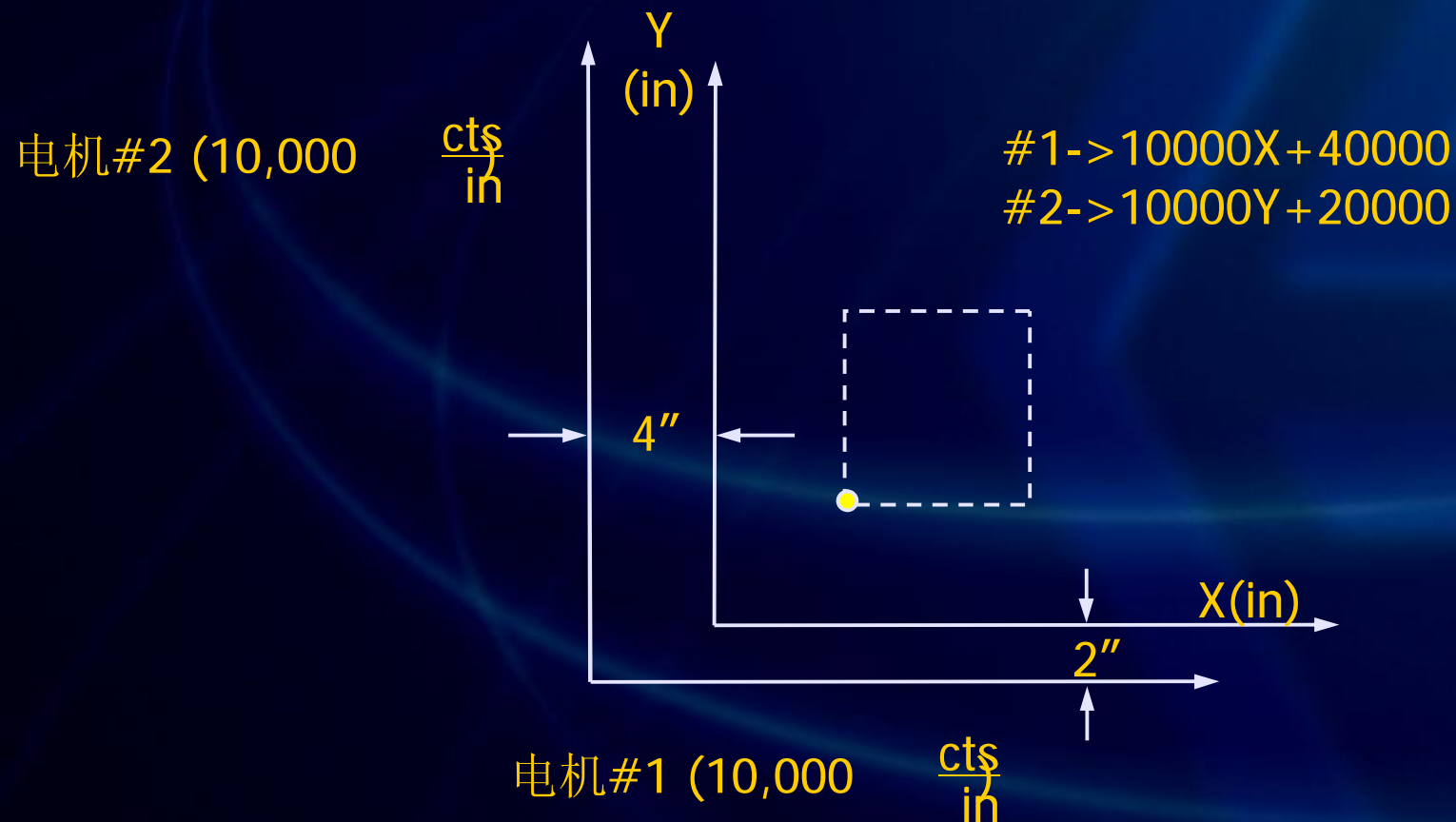
该定义不允许。

当运行两个程序时，一个电机将接收相冲突的指令。

第二个坐标系的定义将被拒绝。

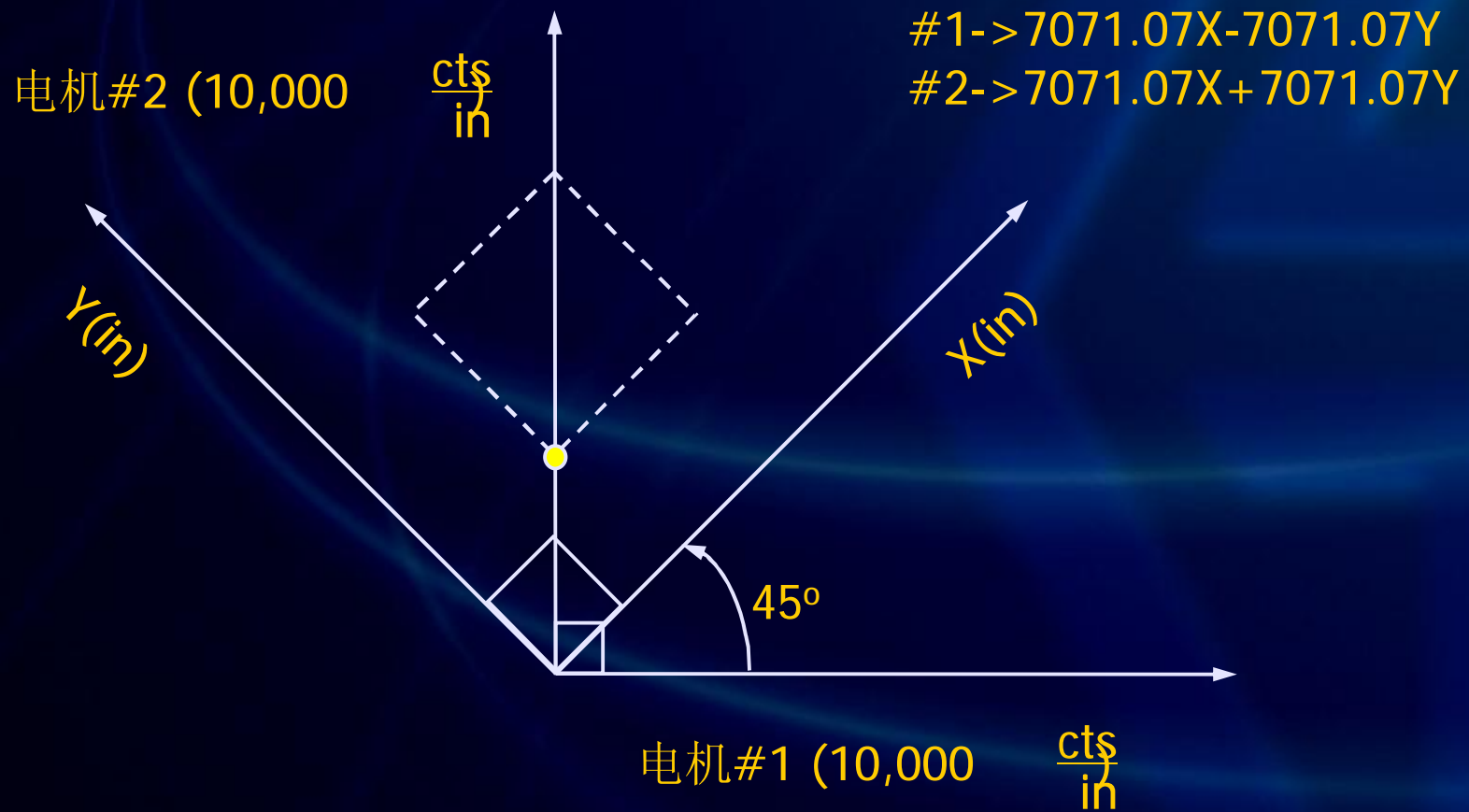
PMAC的坐标定义

- n 比例和平移(每10000个计数对应1英寸)



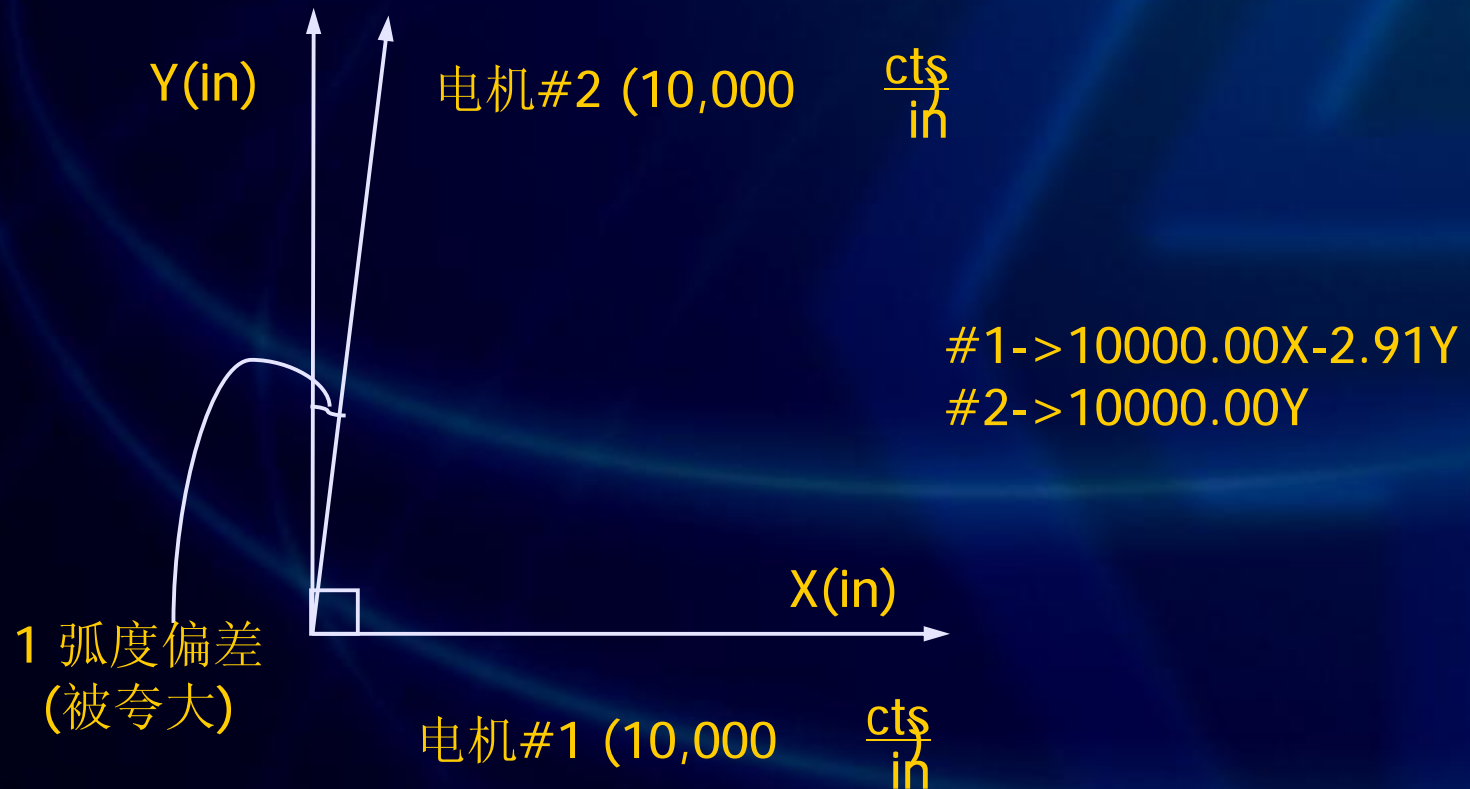
PMAC的坐标定义(续)

n 比例和旋转



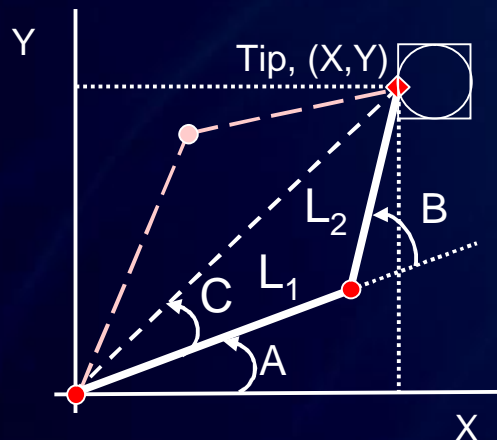
PMAC的坐标定义(续)

n 垂直校正



Turbo PMAC的运动学运算

可以直接编写工具端(Tip)运动



正向运动学

$$X = L_1 \cos(A) + L_2 \cos(A+B)$$

$$Y = L_1 \sin(A) + L_2 \sin(A+B)$$

计算工具端位置
根据 A, B, L₁ & L₂

逆向运动学

$$B = +\cos^{-1} \left(\frac{X^2 + Y^2 - L_1^2 - L_2^2}{2L_1L_2} \right)$$

$$A+C = \text{atan2}(Y, X)$$

$$C = +\cos^{-1} \left(\frac{X^2 + Y^2 + L_1^2 - L_2^2}{2L_1\sqrt{X^2 + Y^2}} \right)$$

$$A = (A+C) - C$$

计算 A & B 角
根据 X & Y 工具端位置

Turbo PMAC中的运动子程序

&1

OPEN FORWARD

CLEAR

Xaxispos = f_1 (Motor1pos, Motor2pos)

Yaxispos = f_2 (Motor1pos, Motor2pos)

CLOSE

OPEN INVERSE

CLEAR

Motor1pos = g_1 (Xaxispos, Yaxispos)

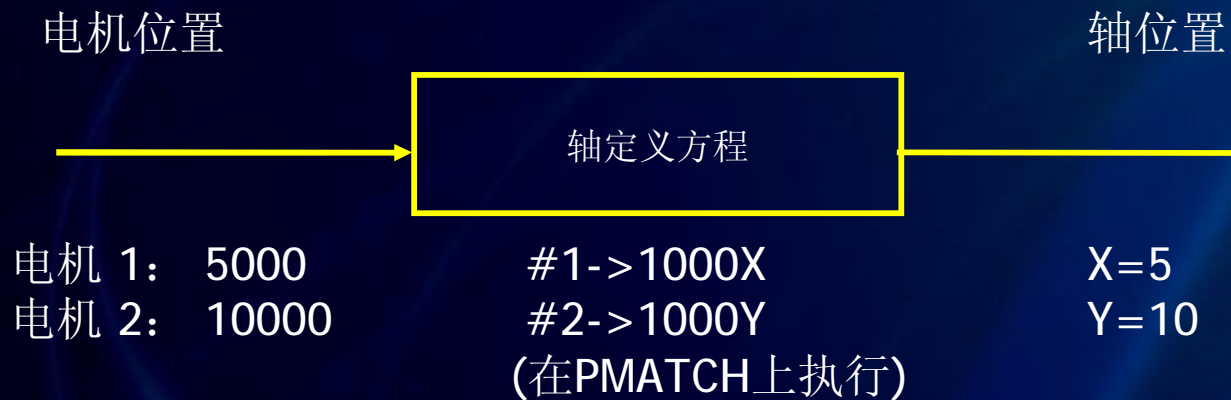
Motor2pos = g_2 (Xaxispos, Yaxispos)

CLOSE

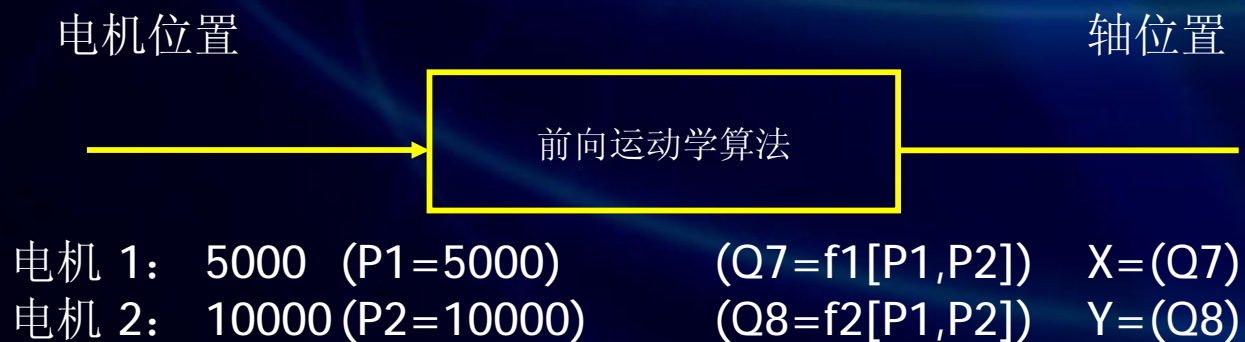
MotorNPos 存储在变量 PN中

A,B,C,U,V,W,X,Y, Zaxispos 存储在坐标系(C.S.)中的变量 Q1-Q9中

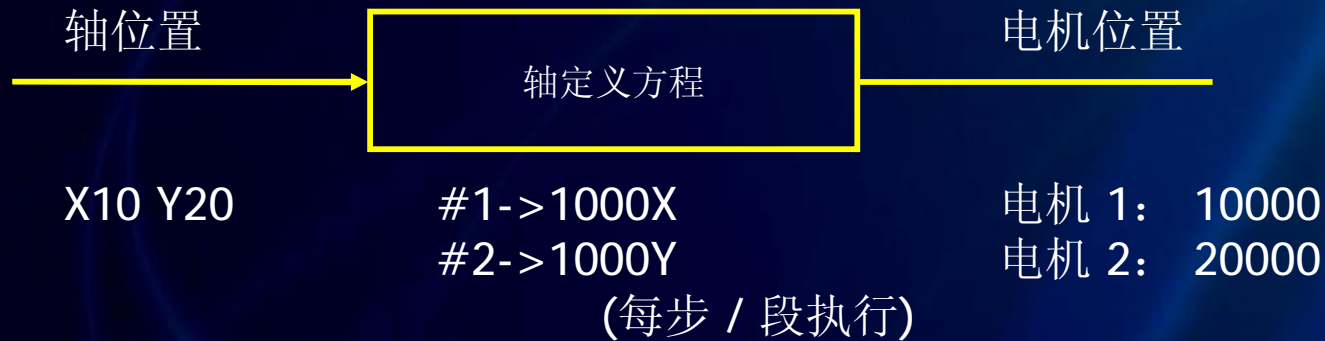
无前向运动学方程的电机到轴的转换



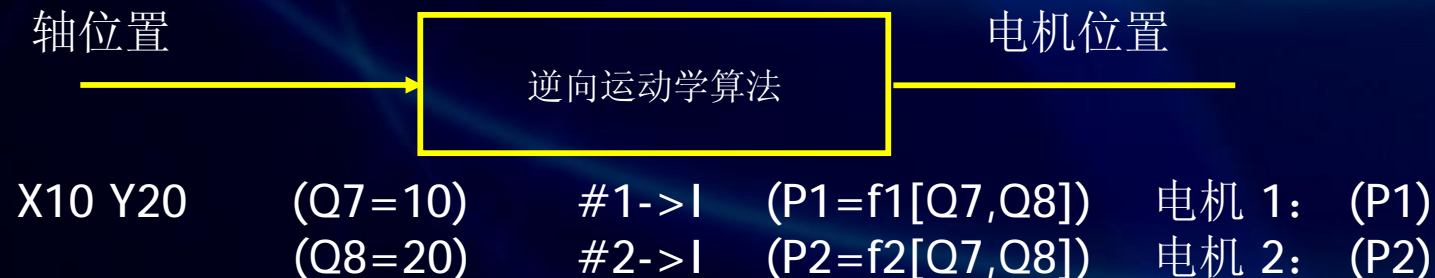
有前向运动学方程的电机-轴的转换



无逆向运动学方程的轴到电机的转换



有逆向运动学方程的轴到电机的转换



PMAC 轴的特性

n X,Y,Z: 传统上的主直线轴

- n 矩阵轴的定义
- n 实时矩阵轴的转换
- n 圆弧插补
- n 刀具半径补偿

n U,V,W: 传统上的辅直线轴

- n 矩阵轴的定义

n A,B,C: 传统上的旋转轴

- n (A 围绕 X, B 围绕 Y, C 围绕 Z)
- n 位置翻转(1xx27)

可编程的轴偏移量

- n 由于编程的目的轴的原点可以被改变
- n 可以采用重新定义当前的命令位置来改变
- n 在线命令 {轴}={constant}
例. X=10 Y=20
- n 缓冲区运动程序命令 PSET {轴}{数据}...
例. PSET X10 Y(P1)
- n 这些命令改变电机原点和轴原点之间的寄存器“位置偏移”
- n 也可直接写入位置偏移寄存器 (然后用 PMATCH 命令去重新对齐电机和轴)

轴的矩阵转换

n 从基坐标变换:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} Q(n) & Q(n+1) & Q(n+2) \\ Q(n+3) & Q(n+4) & Q(n+5) \\ Q(n+6) & Q(n+7) & Q(n+8) \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} Q(m) \\ Q(m+1) \\ Q(m+2) \end{bmatrix}$$

旋转矩阵

位移矢量

AROTn : 绝对旋转
ADISm : 绝对位移

轴矩阵的转换(续)

n 从当前的转换坐标变换:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} Q(n) & Q(n+1) & Q(n+2) \\ Q(n+3) & Q(n+4) & Q(n+5) \\ Q(n+6) & Q(n+7) & Q(n+8) \end{bmatrix} \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} + \begin{bmatrix} Q(m) \\ Q(m+1) \\ Q(m+2) \end{bmatrix}$$

旋转矩阵

位移矢量

IROTn : 增量旋转
IDISm : 增量位移

使用转换矩阵

- n 在线命令以保留内存:

DEF TBUF {# of matrices} ;定义矩阵的数目

- n 运动程序命令:

TSELECT{xform} ; 选择使用的转换

; TSELECT0禁用矩阵计算

TINIT ; 确保选用的变换为要使用的变换

{Q-变量 = values}

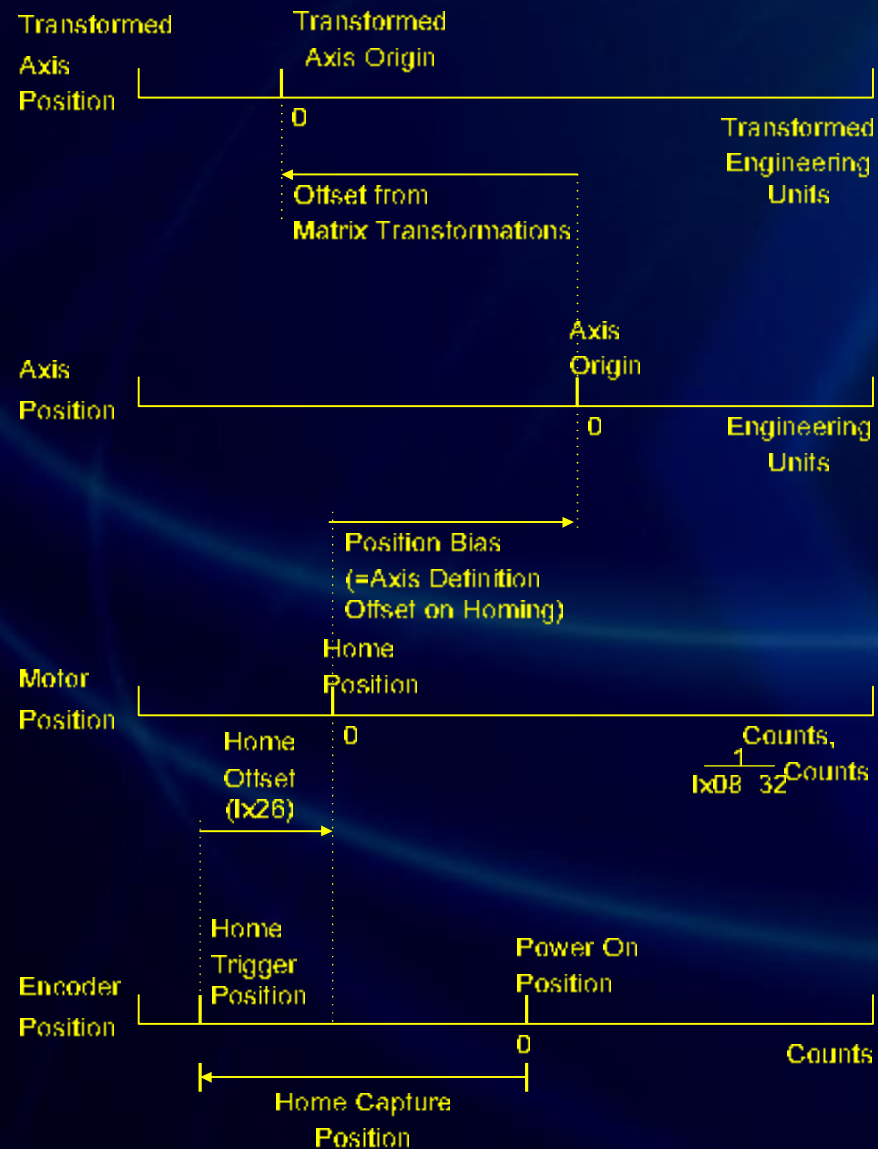
AROTn ; 将Q(n) - Q(n+8)放入旋转矩阵

IROTn ; 将旋转矩阵 与Q(n) - Q(n+8) 矩阵相乘

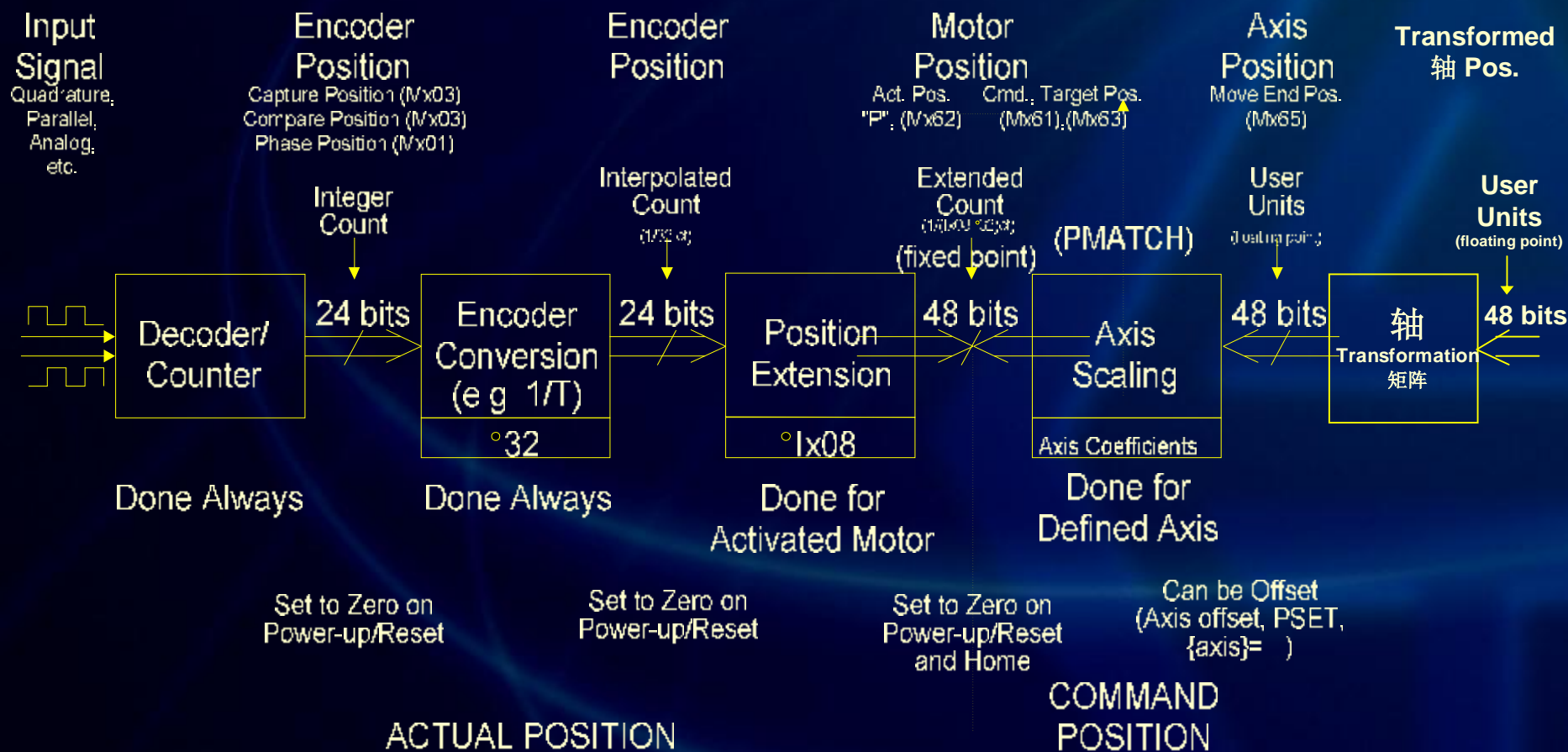
ADISm ; 将位移向量放入 Q(m) - Q(m+2)

IDISm ; 将 Q(m) - Q(m+2) 加到位移矢量

PMAC的位置比例



PMAC的位置处理



位置跟踪 (主/从)

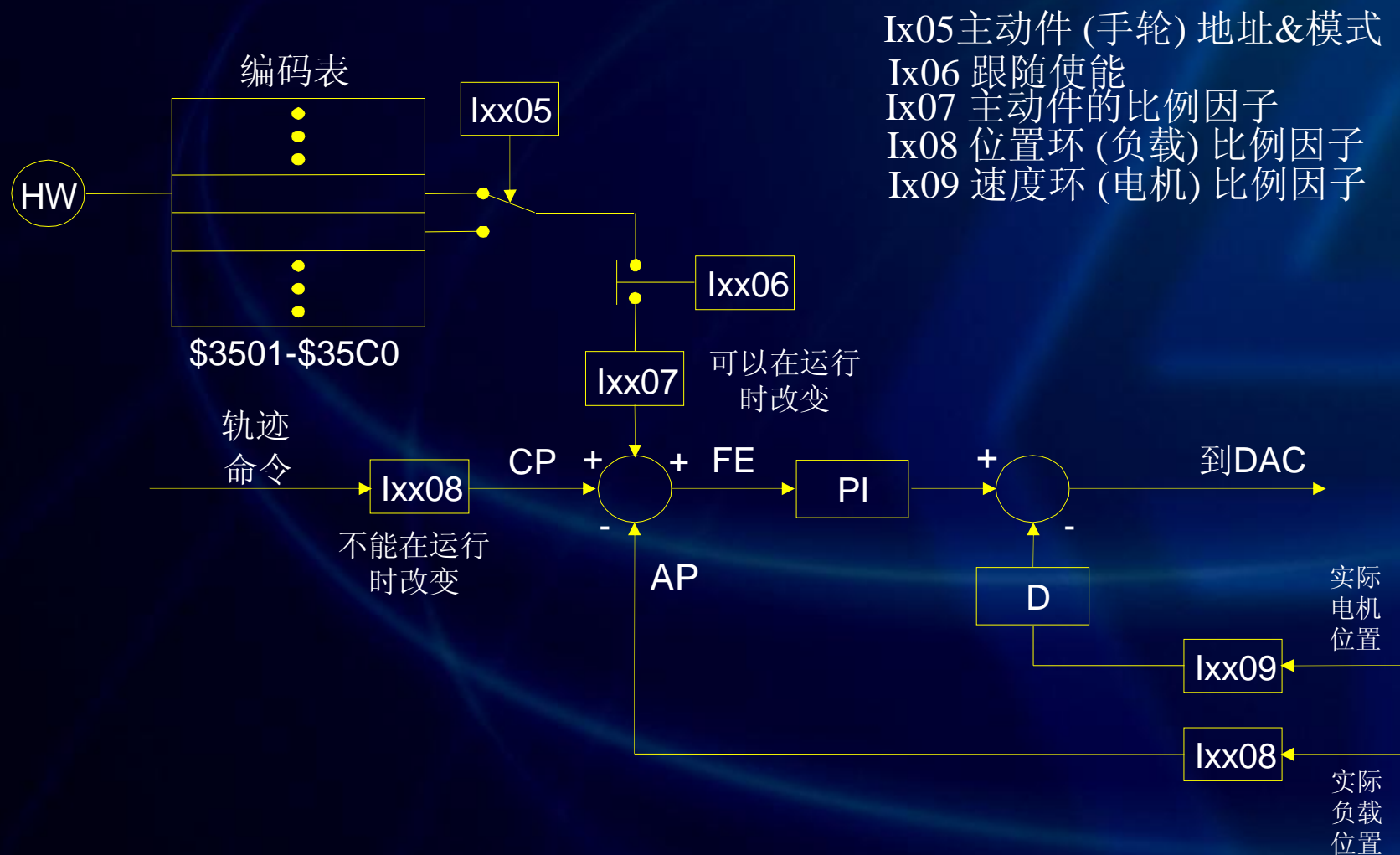
地址 I-变量 (续)

n Ixx05 主动件的(位置)地址

用于位置跟随 (电子齿轮)
为主动件指定(xx) 寄存器

n Ixx06 位 0 跟随启用/禁用
 位 1 指定正常/偏移方式跟随
 跟随不会改变报告的位置;
 程序运动可以叠加

位置跟随参数



Ix05主动件 (手轮) 地址&模式

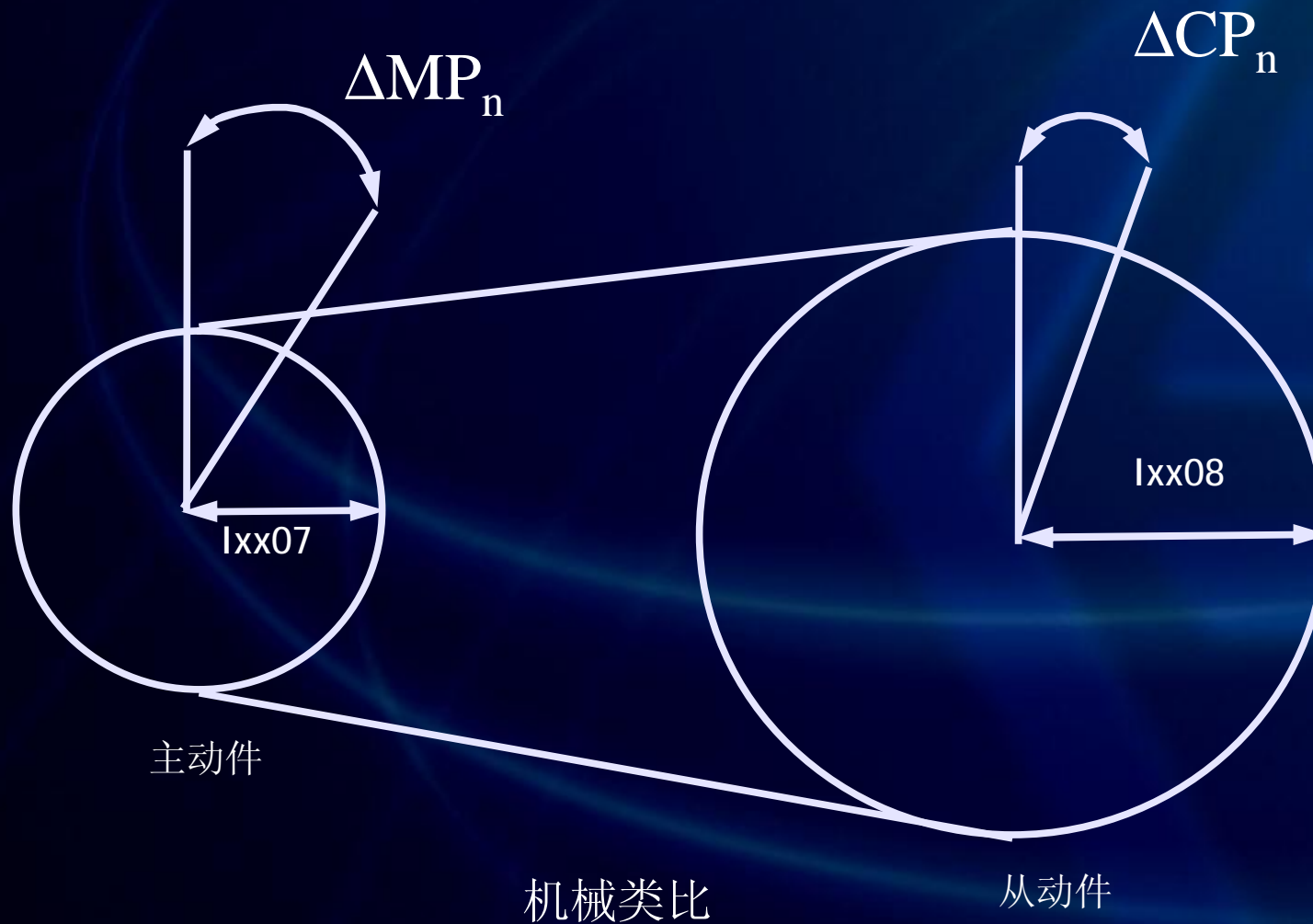
Ix06 跟随使能

Ix07 主动件的比例因子

Ix08 位置环 (负载) 比例因子

Ix09 速度环 (电机) 比例因子

PMAC 位置跟随 (电子齿轮)



PMAC的位置跟随

电机
命令
位置 } 跟随 { 主动件
位置

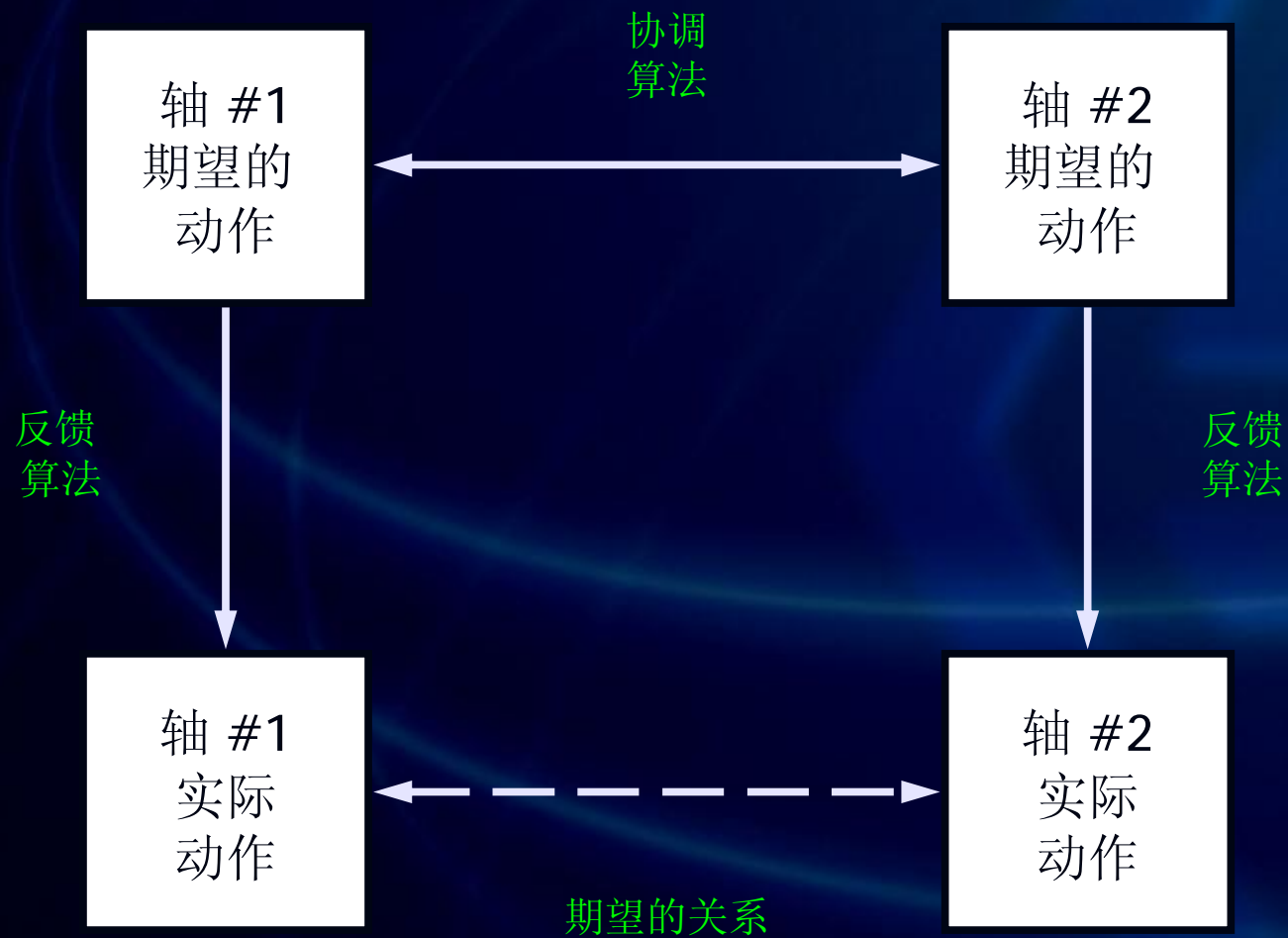
$$\Delta CP_n = \frac{Ix07}{Ix08} * \Delta MP_n$$

$$Ix08 * \Delta CP_n = Ix07 * \Delta MP_n$$

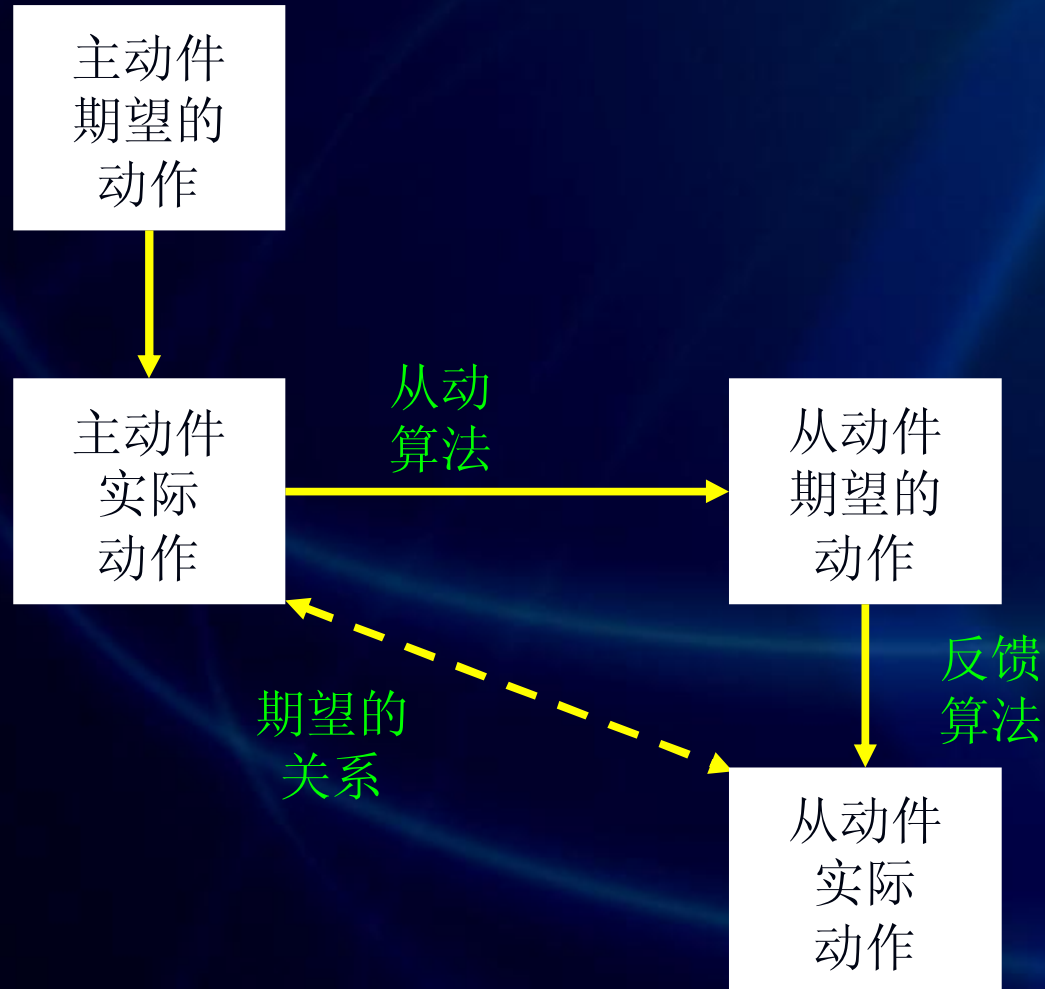
With 1/T:

$$32 * Ix08 * \Delta CP_n = 32 * Ix07 * \Delta MP_n$$

协调概念



主 - 从概念



下列场合使用协调功能

- n 所有的轴都能被很好的控制时
- n 在功能上，没有那一轴作为主要轴
 - n 如果不能确定哪个为主动件，采用协调！
- n 容易描述同步运动时
 - n 例如：同时启动、停止、转弯

使用主-从当:

- n 轴的紧密控制代价太高时
 - n 例如: 大功率主轴或者输送机电机
- n 轴的紧密控制太难时
- n 编写主/从运动程序相对简单时(这里推荐使用从动到命令位置)
 - n 例如: 线圈绕组, 螺纹加工

主/从实例操作

- n 在演示软件中，设置电机#2为电机#1的从动电机，然后手动电机1.
- n 对电机#2，使用变量lxx07尝试改变不同的主/从率。

运动程序

PMAC的运动程序

- n 自动的执行序列运动
- n 执行运动时同时能执行数学和逻辑功能 (在运动边缘)
- n 运动语句像机床上用的 "G 代码"
- n 数学和逻辑语句像BASIC语言
- n 通过坐标系运行程序
- n 轴命令采用工程单位

PMAC 运动程序的语句

n 运动指令

X1000 Y2000 Z3000

U(P1*3.14159) V(20*SIN(Q6))

DWELL, DELAY

n 模块指令

ABS, INC, FRAX, NORMAL

LINEAR, RAPID, CIRCLEn, SPLINEn, PVT

TA, TS, TM, F

n 变量赋值

{变量} = {表达式}

运动程序的逻辑控制

- n 条件分支:
IF ({条件})
[ELSE]
ENDIF
- n 循环:
WHILE ({条件})
ENDWHILE
- n 标号:
N {常数}
O {常数}
- n 子程序:
GOSUB {数值}
CALL {数值}
G, M, T, D
RETURN
- n 跳转:
GOTO {数值}

PMAC的逻辑操作符

n 逻辑操作符

&	(位比较“与”)
	(位比较“或”)
^	(位比较“异或”)

n 比较符

=	(等于)
!=	(不等于)
>	(大于)
!>	(不大于; 小于或等于)
<	(小于)

n 函数

SIN, COS, TAN, ASIN, ACOS, ATAN, ATAN2,
SQRT, LN, EXP, ABS, INT

运动程序的运行流程

n 下一个实时中断产生后，在以下动作发生后开始程序运算：

- 1) 接收到运行(Run)指令
- 2) 当运动执行(插值)到一个新的块时，块请求标志被置位

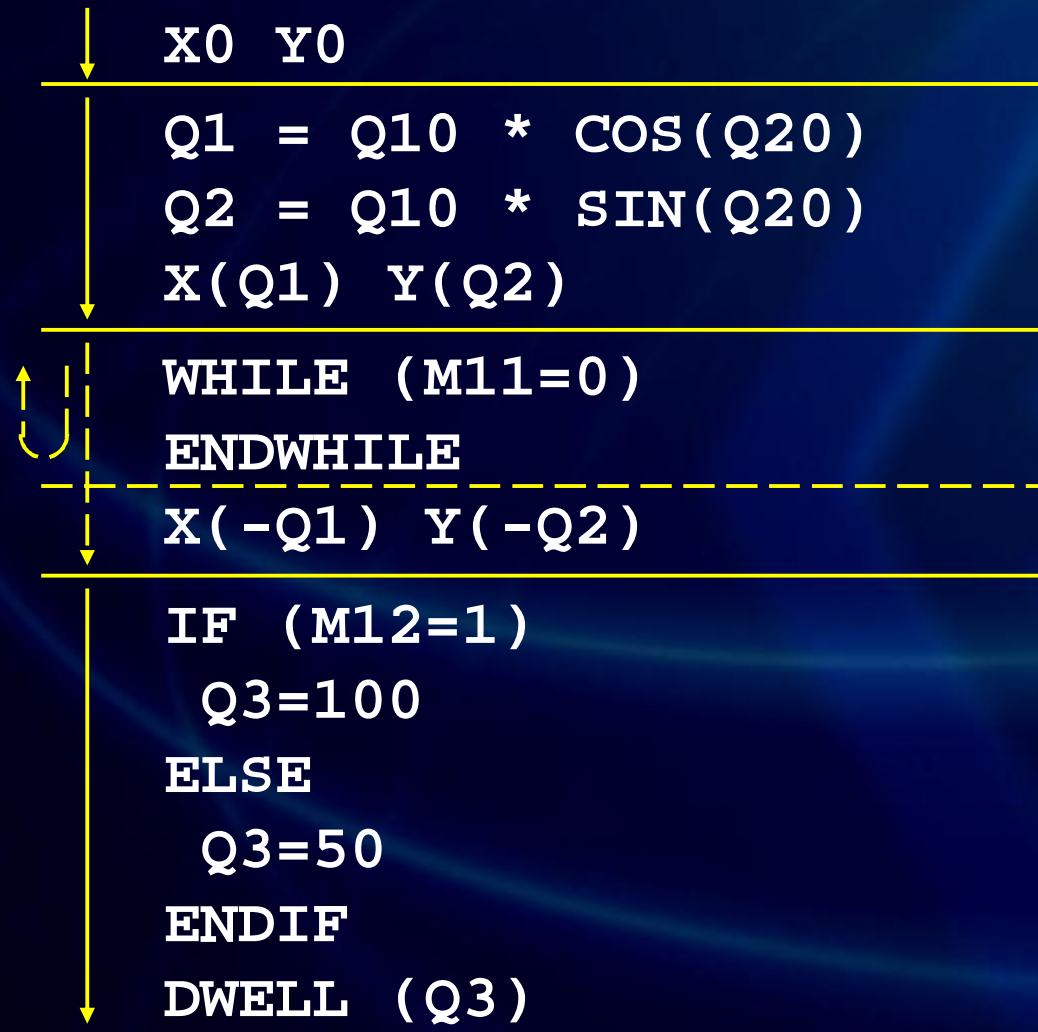
运动程序的运行流程(续)

继续程序运算，直到：

- 1) 计算下一个运动，或发现DELAY或DWELL指令
(清除块请求标志；在下一个块申请标志后继续计算)
- 2) 在如下程序后跳回两步
 - a) ENDWHILES (包括隐含的情况)
和/或
 - b) GOTO's to higher lines

(块请求标志仍旧被留为置位状态；
禁用混合；
运动停止后继续计算；
如果已经停止，则在下一个实时中断后继续)

运动程序流程的示例



观测PMAC的“两次回跳”规则

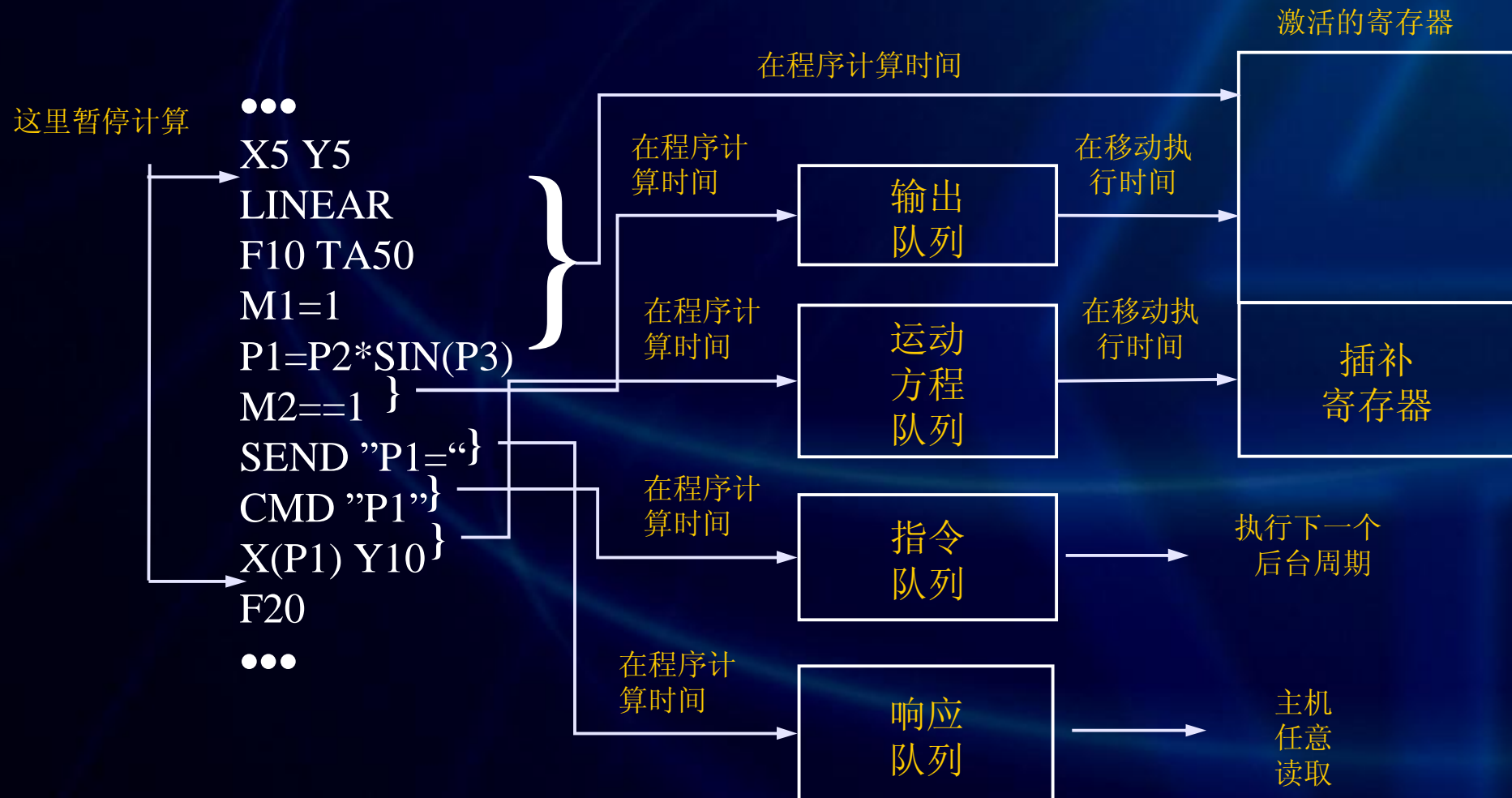
```
P1=1
WHILE (P1<11)
  P2=0
  WHILE (P2<360)
    P3=10+P1*COS(P2)
    X(P3)
    P2=P2+10
  ENDWHILE
  P1=P1+1
ENDWHILE
```

(每次退出内环时混合停止：在下次运动前，两个Endwhile语句相遇)

```
P1=1
WHILE (P1<11)
  P2=0
  WHILE (P2<350)
    P3=10+P1*COS(P2)
    X(P3)
    P2=P2+10
  ENDWHILE
  P3=10+P1*COS(P2)
  X(P3)
  P1=P1+1
ENDWHILE
```

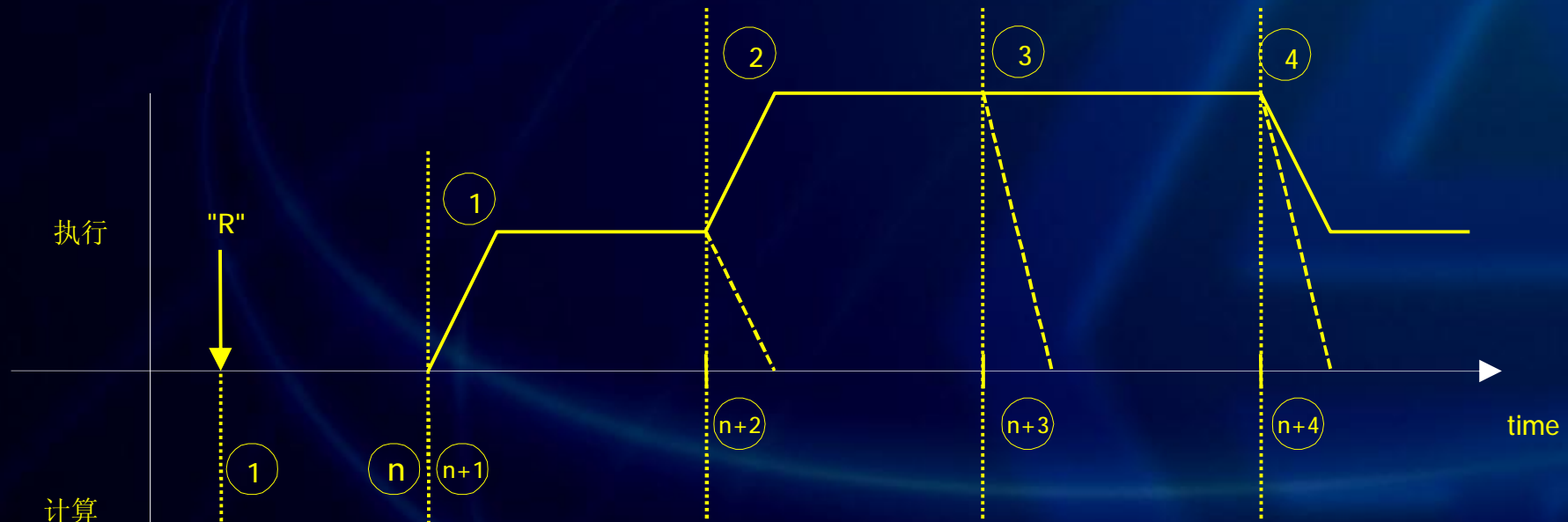
(整个例子运行过程混合)

PMAC 运动程序执行



PMAC 运动程序的超前计算

n 混合运动提前“n”步

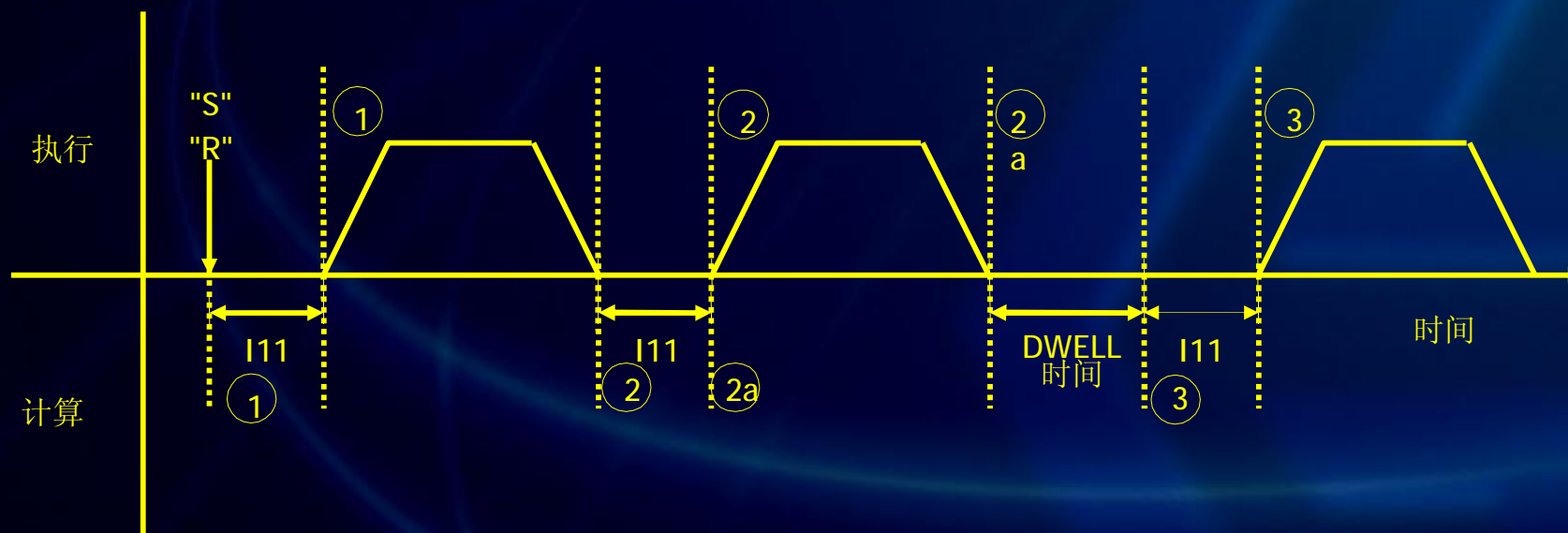


“n”有多大？

- 点对点运动为0, 无半径补偿
- 混合运动加1
- 基本的加速度控制加1
- 刀具半径补偿加1
- 为了稳定的加速度控制, “n”尽可能的大, 以覆盖停止距离

PMAC 运动程序的超前计算(续)

- n 在 $Isx92=1$, 快速、步进、停和步进(RAPID, HOME, DWELL, "S") 前无前瞻



例1：简单运动

本例将教你怎样在PMAC上编写一个简单的运动程序；首先程序指定了怎样运动，然后执行运动。

***** 设置和定义 *****

```
DEL GAT ; 清除所有定义的采样缓存
&1      ; 坐标系1
CLOSE   ; 确保所有的程序缓存关闭
#1->X   ; 电机#1分配到x-轴 - x轴的一个
        ; 单位就是一个编码计数
```

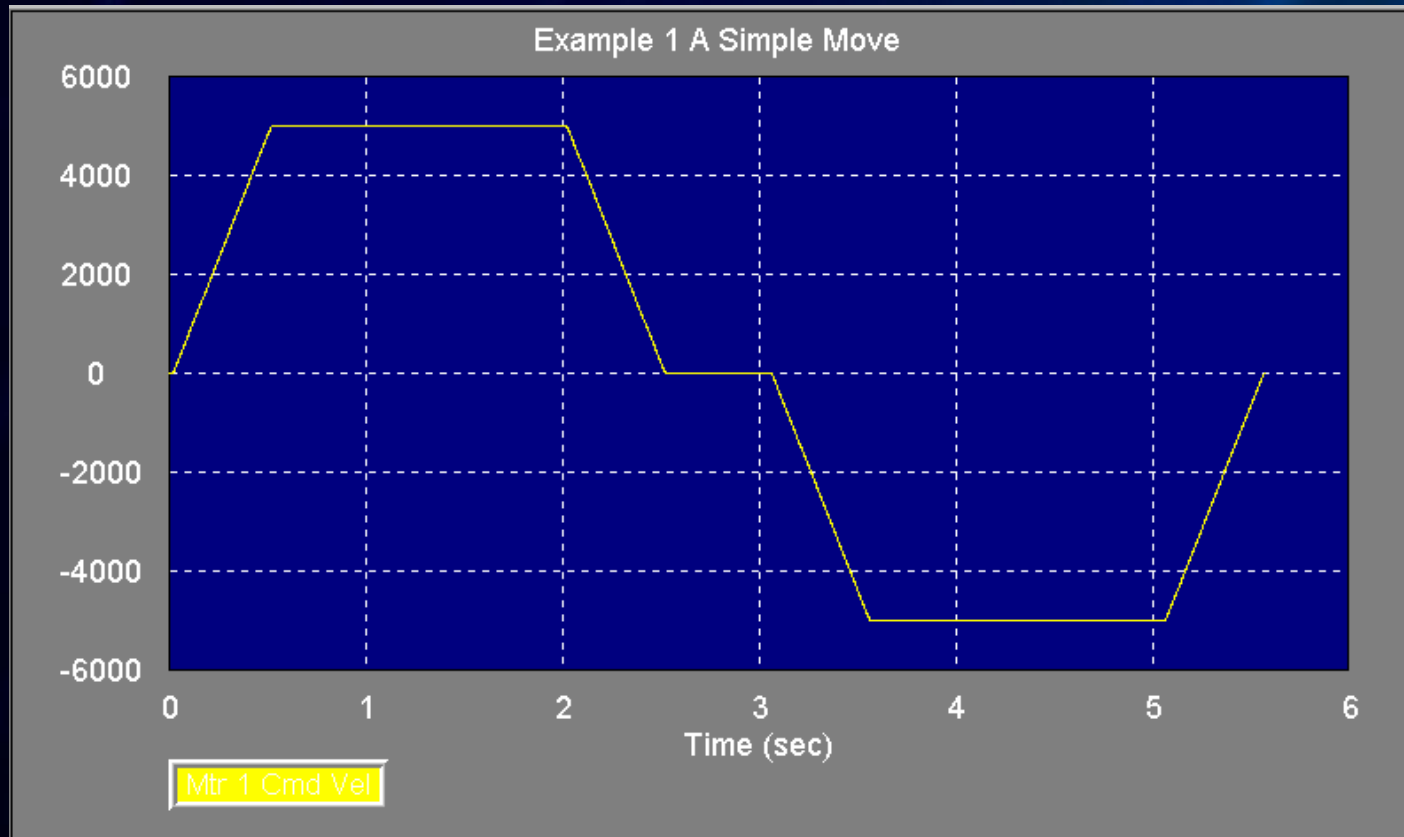
***** 运动程序文本 *****

```
OPEN PROG 1 ; 打开缓存，打开1号运动程序的缓冲入口
CLEAR        ; 清除缓存内容
LINEAR       ; 直线插补运动方式
ABS          ; 绝对值方式 - 运动由位置规定
TA500        ; 设置1/2秒(500毫秒)加速时间
TS0          ; 设置无s-曲线加速时间
F5000        ; 设置5000单位(cts)/sec的进给速率(速度)
X10000       ; 使x-轴运动到位置10000
DWELL500     ; 在此停顿1/2秒(500毫秒)
X0           ; 使x-轴运动到位置0
CLOSE        ; 关闭缓存 - 结束程序
```

运行此程序：

```
&1 B1 R ; 寻址坐标系1，指向程序1开始，运行
```

例1：简单运动(续)



例2：较复杂的运动

- n 本例介绍增量式定时运动和的运动，循环逻辑，变量使用，轴缩放，以及简单的数学计算。注意逻辑和数学计算运动不会给运动带来延迟。

```
***** 设置和定义 *****
```

```
&2          ; 坐标系2  
            ; 注：一个电机不能同时定义在  
            ; 多个坐标系里
```

```
CLOSE       ; 确保所有的缓存关闭
```

```
#5->1000X   ; 电机5的X轴的一个单位(cm)计数是1000个计数
```

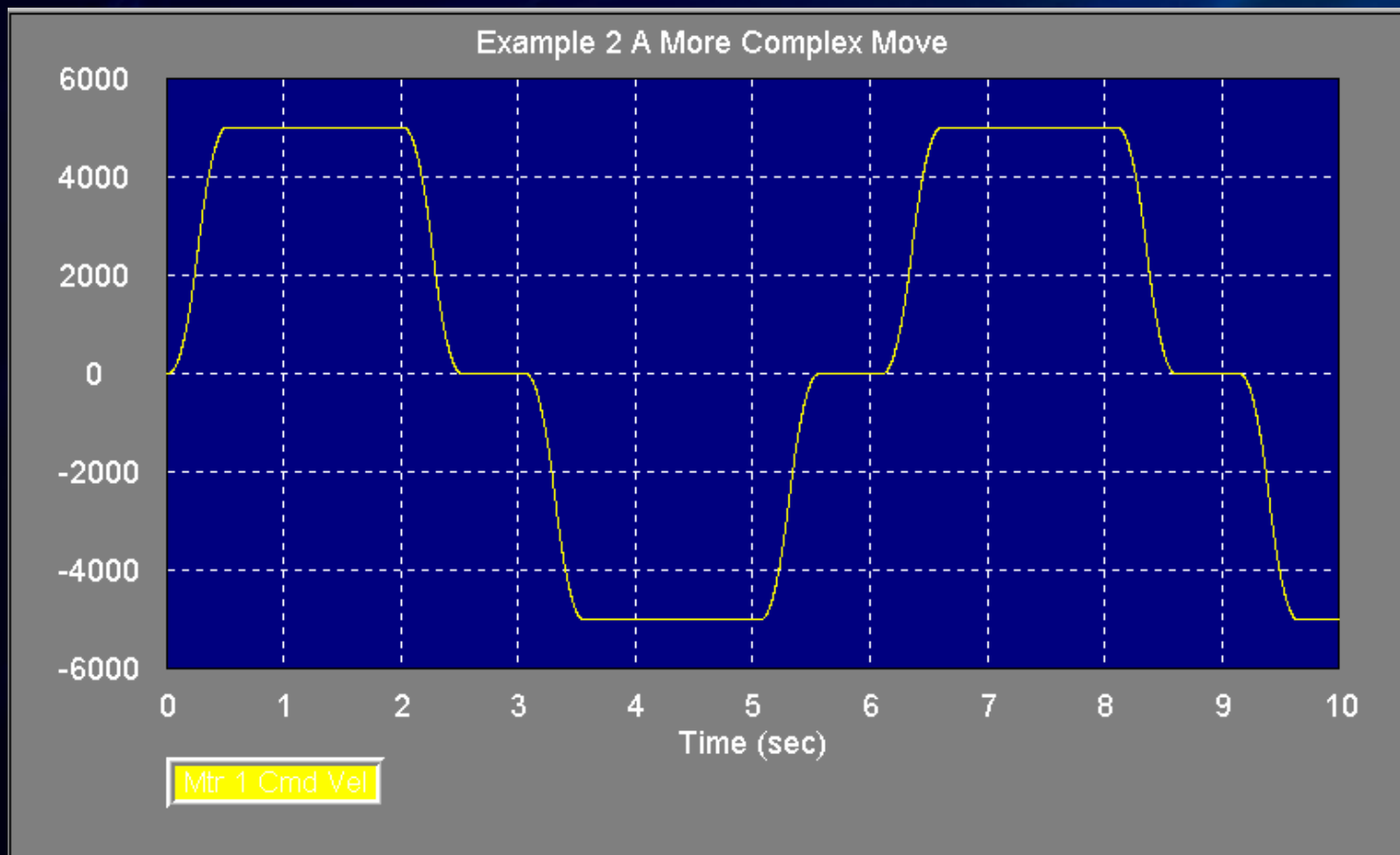
例2： 较复杂的运动(续)

```
;***** 运动程序文本 *****
OPEN PROG 2      ; 打开2号程序的缓存入口
CLEAR            ; 清除缓存内容
LINEAR           ; 直线插补运动方式
INC              ; 增量模式-运动量由距离表示
TA500            ; 1/2秒(500毫秒)加速时间
TS250            ; 每半个s-曲线为1/4秒
P1=0             ; 初始化循环计数变量
WHILE (P1<10)    ; 循环直到条件为假(10次)
    X10          ; x-轴正向运动10cm(=10,000 cts)
    DWELL500     ; 停止1/2秒
    X-10         ; x-轴反向运动10cm
    DWELL500     ; 停止1/2秒
    P1=P1+1      ; 计数器加 1
ENDWHILE         ; 循环结束
CLOSE            ; 关闭缓存-结束程序
```

运行此程序:

```
&2 B2 R          ; 寻址坐标系2, 指向程序2开始, 运行
```

例2：较复杂的运动(续)



旋转运动程序缓冲区(Rotary Motion Program Buffers)

n 可以实时下载程序

- n **DEFINE ROT {常数}** 为寻址的坐标系缓存保留指定一定数目双字节内存作为旋转运动程序缓冲区
- n **B0** 告知坐标系准备执行旋转缓存。(使用这个命令时所有缓冲区必须关闭)
- n **OPEN ROT** 为输入打开所有的旋转运动程序缓冲区
- n **CLEAR** 擦除所有存在的内容

旋转运动程序缓冲区(续)

- n 该缓存中采用命令行的方式输入
- n **R** 命令让被寻址的坐标系开始执行运动程序
- n **PR** 报告该坐标系中已经被输入旋转缓存但还没有执行(剩余程序)的程序行数且
- n **I2=1或3**时，对于寻址的坐标系，**<BREQ>** 中断在程序执行到**I17**行时产生；到**I16**行时再产生一次；对于选址的左边系，
- n **CLOSE** 关闭当前打开的缓存；但不停止执行程序

运行“裸”旋转缓冲

- n 旋转缓冲已经执行了所有发送给它的运动，处于无限“等待”状态
- n 将马上执行新发送给它的运动
- n 使“缓冲”程序命令的执行如在线命令一样
- n 由状态位决定运动的完成

PMAC 运动模式

n 直线(LINEAR) 模式

- n 梯形/三角形速度曲线
- n 笛卡尔坐标系中的直线路径
- n 与LINEAR和CIRCLE运动可混合

n 圆弧(CIRCLE)模式

- n 正弦速度曲线
- n 笛卡尔坐标系中的圆弧路径
- n 与LINEAR + CIRCLE运动可混合

PMAC 运动模式(续)

n 快速(RAPID) 模式

- n 梯形/三角形速度曲线
- n 笛卡尔坐标系中的(近似)直线路径
- n 点对点最小时间；无混合

n 样条(SPLINE) 模式

- n 二次速度曲线
- n 立方 B-样条路径
- n 与SPLINE运动可混合，保持加速度曲线连续

n 位移速度时间(PVT) 模式

- n 二次速度曲线
- n Hermite样条路径
- n 可与PVT运动混合，保持速度曲线连续

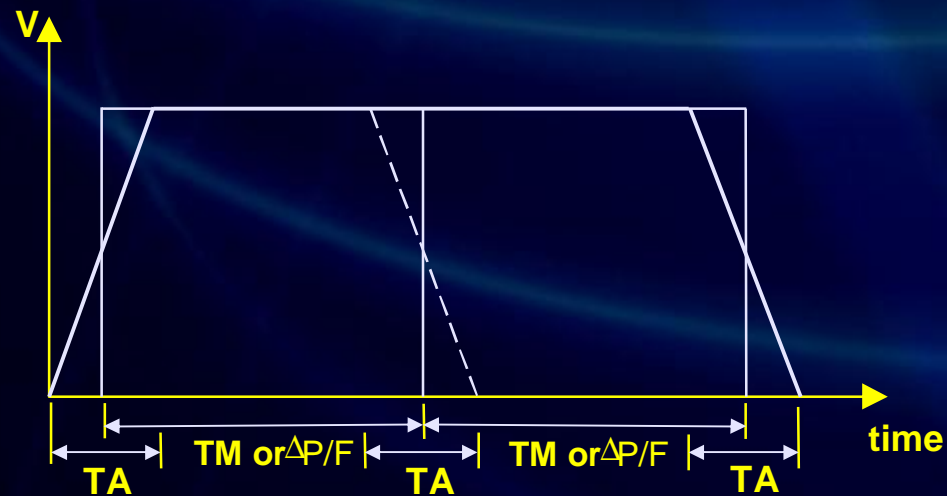
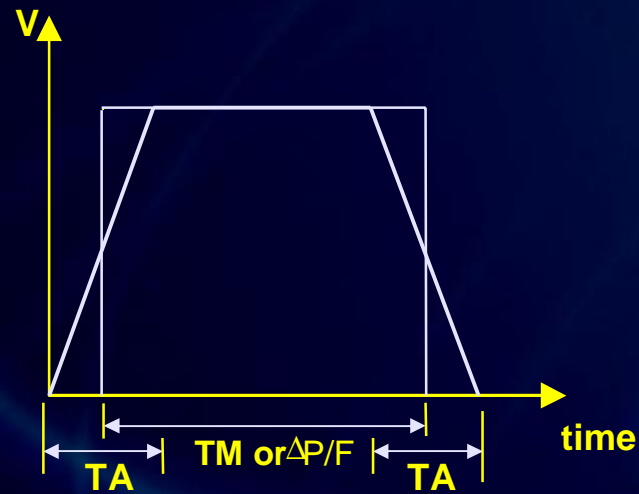
线性运动模式

线性模式运动

- n 需要指定终点 (ABS) 或距离(INC)
- n 指定矢量速度(F)或运动时间(TM)
- n 指定加速时间(TA, TS) 或速率 (Ix17)
- n F 用于指定矢量速度大小，与运动的方向无关
- n 当 $TM=TA$ ， $TS=0$ 时，相当于二次 B-样条路径

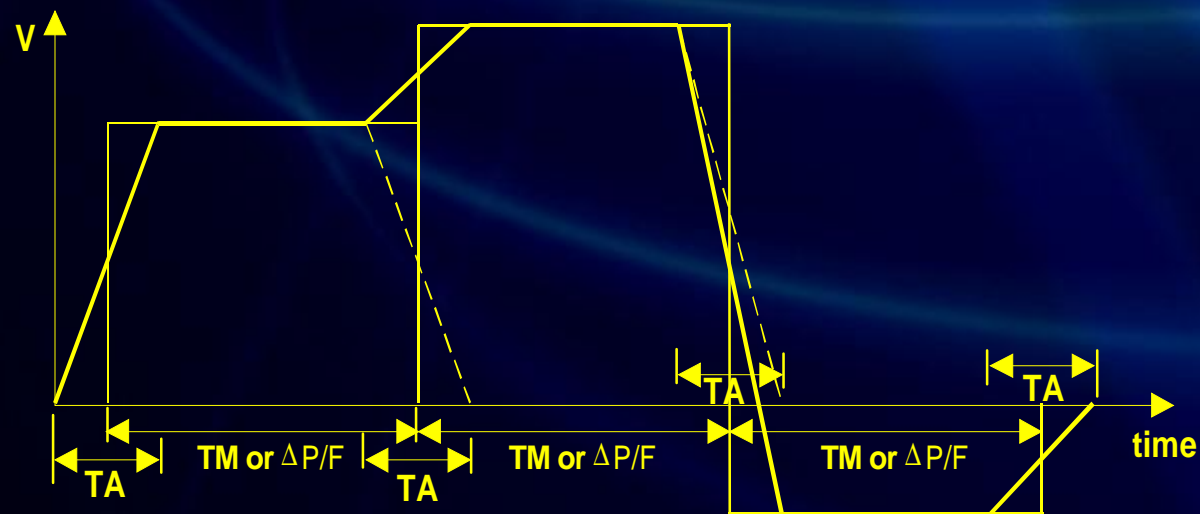
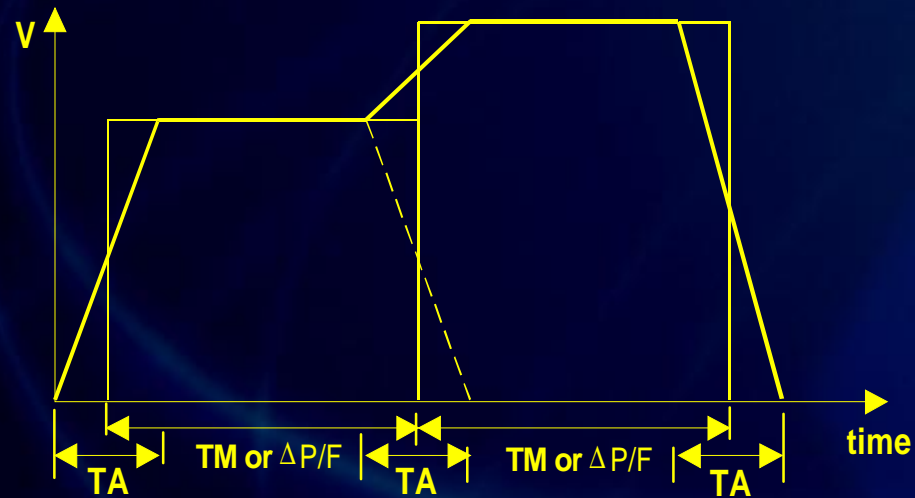
线性模式轨迹

n 小加速时间



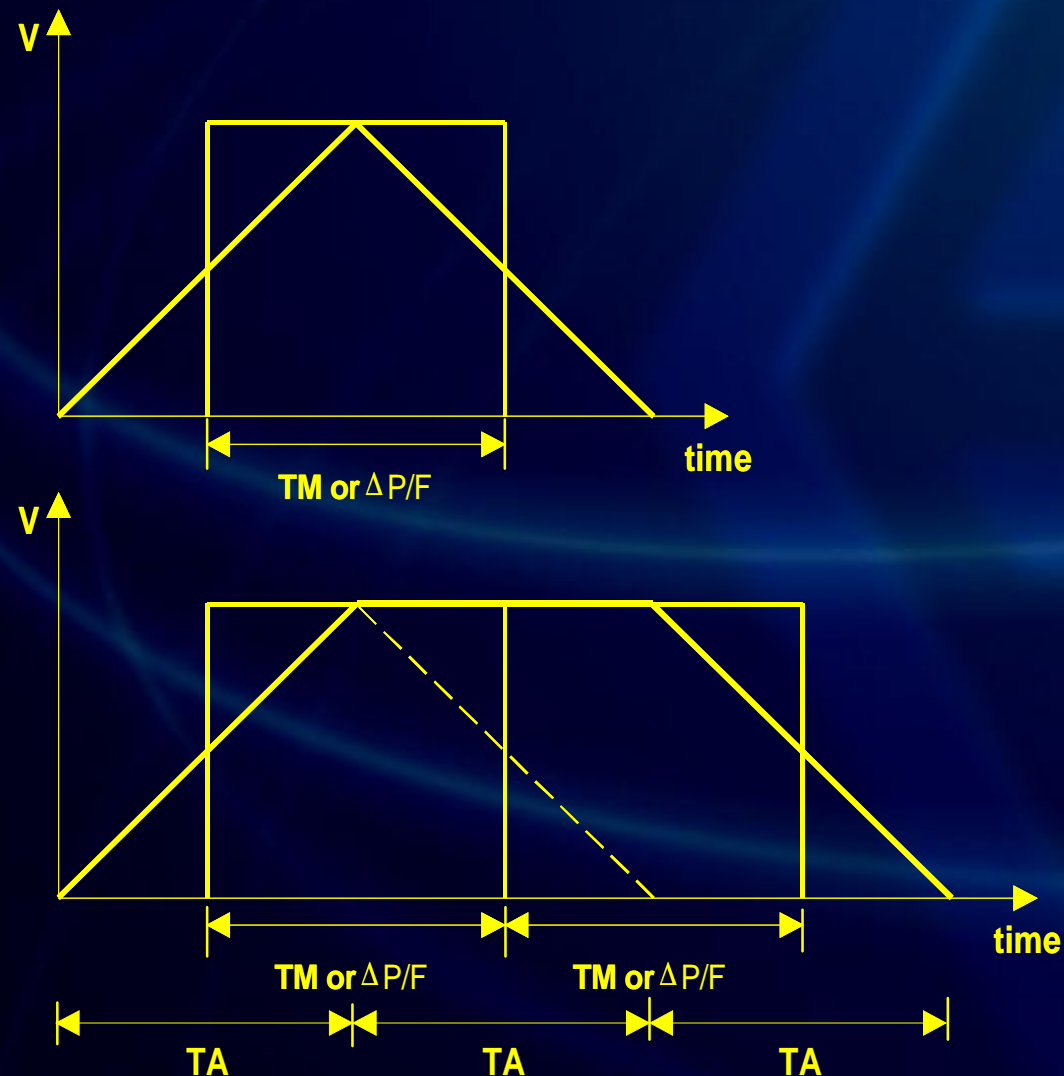
线性模式轨迹(续)

n 小加速时间

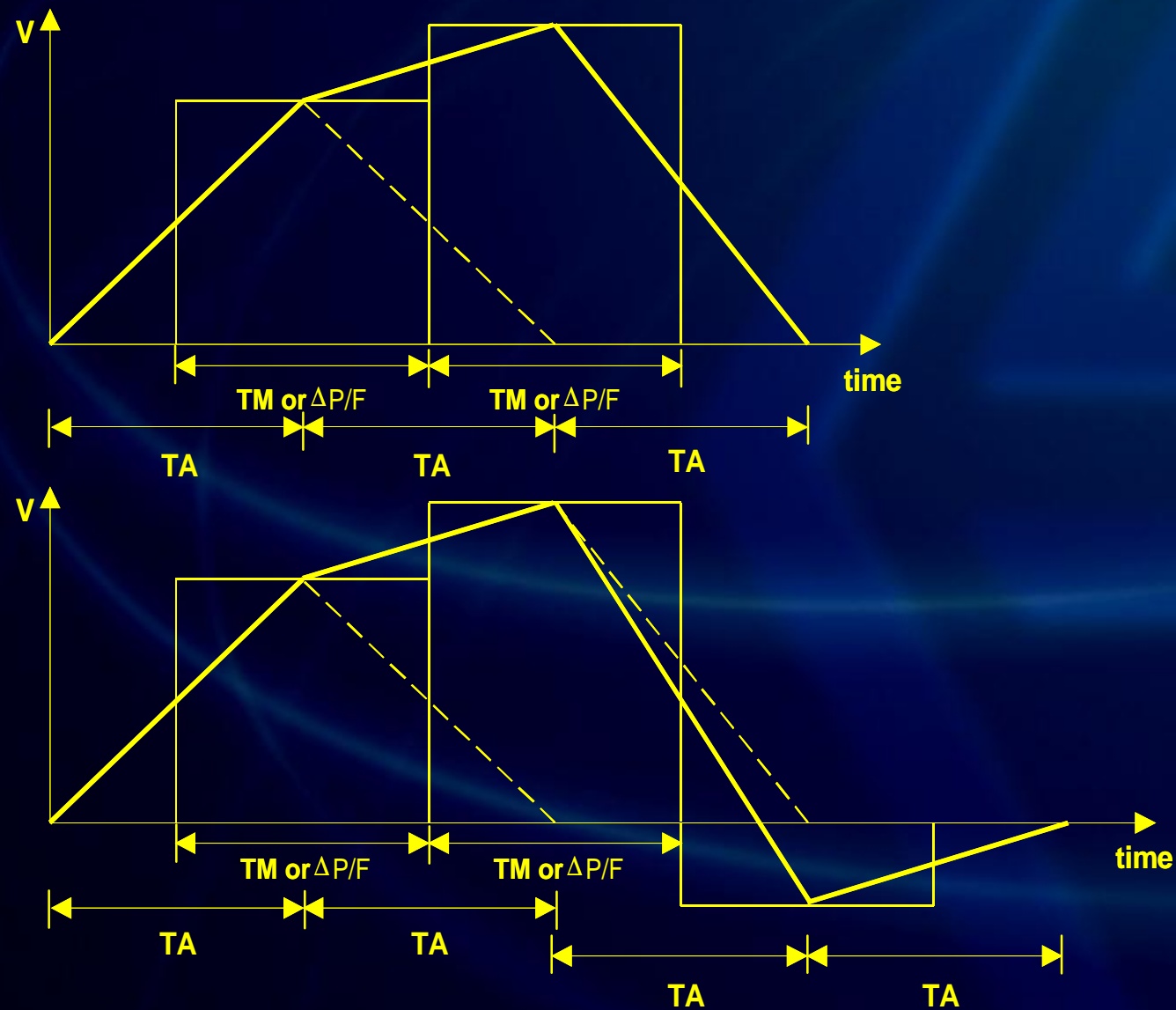


线性模式轨迹(续)

n 加速时间与运动时间匹配

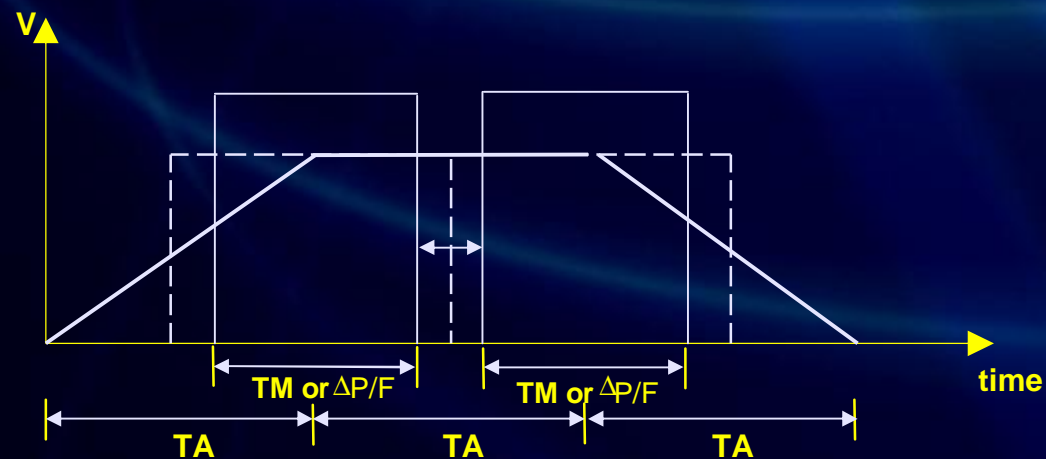
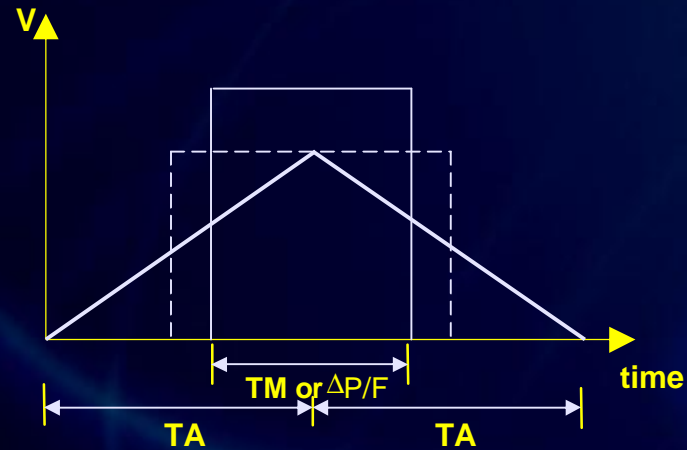


线性模式轨迹(续)



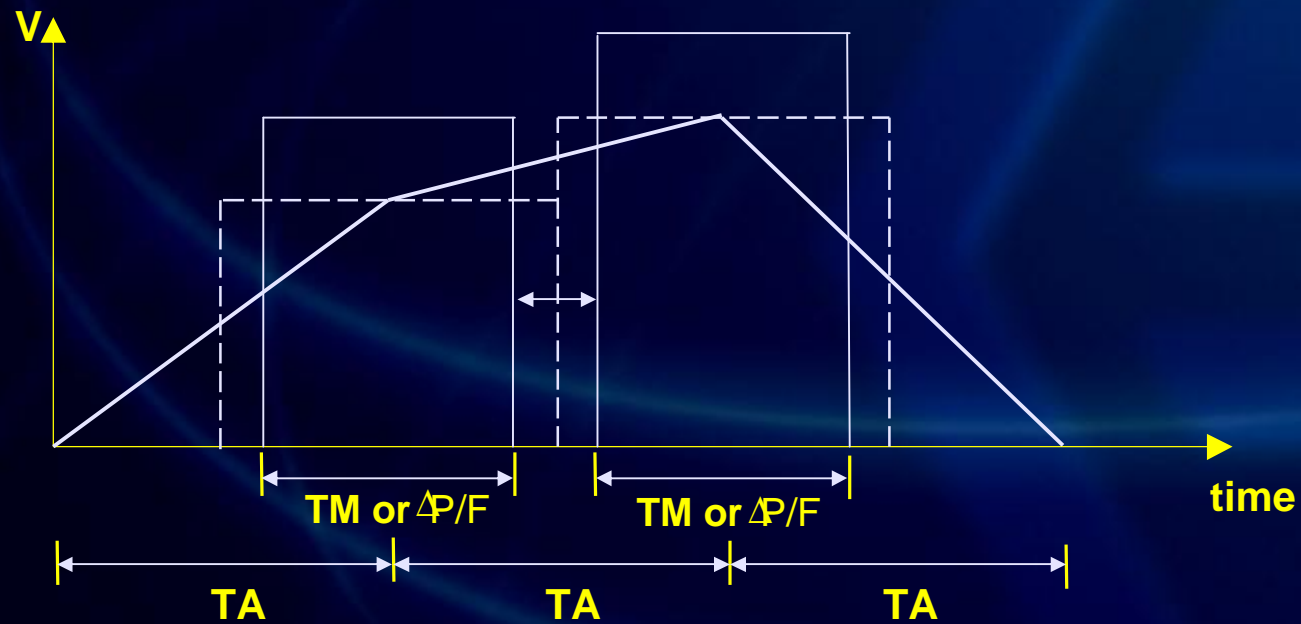
线性模式轨迹(续)

n 大(速度限制) 加速时间



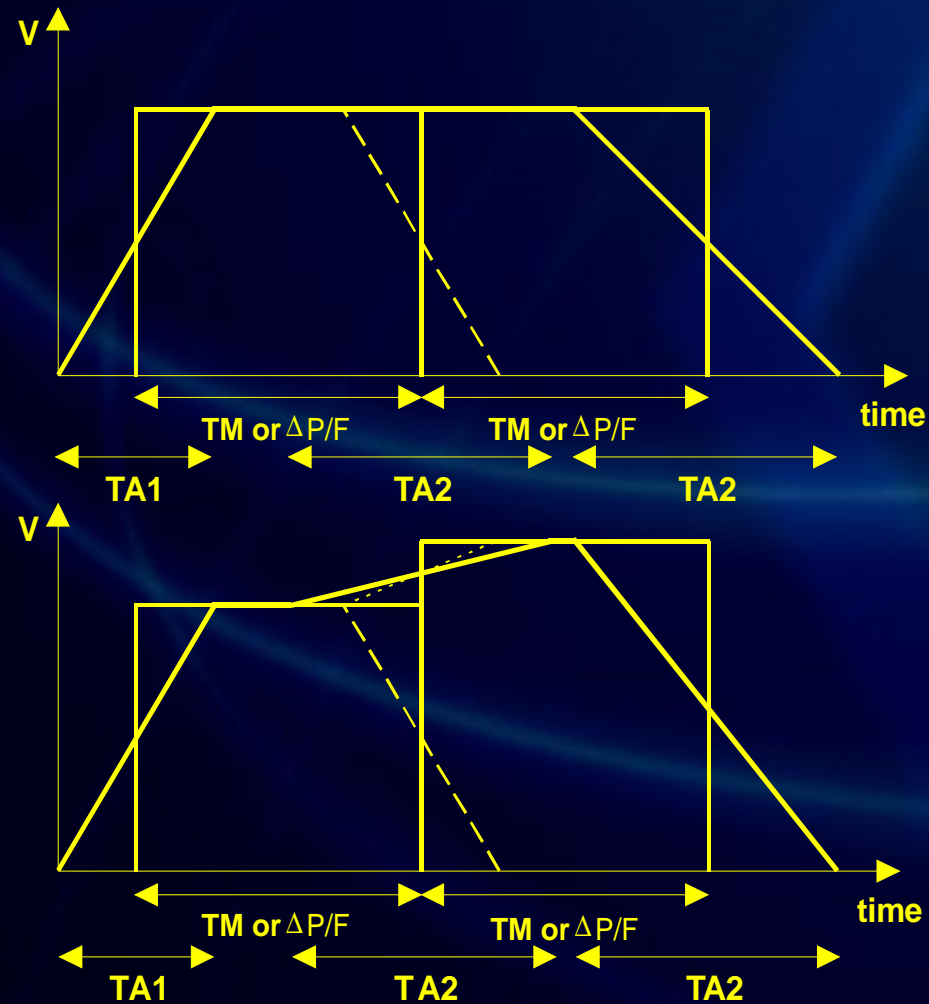
线性模式轨迹(续)

n 大(速度限制) 加速时间



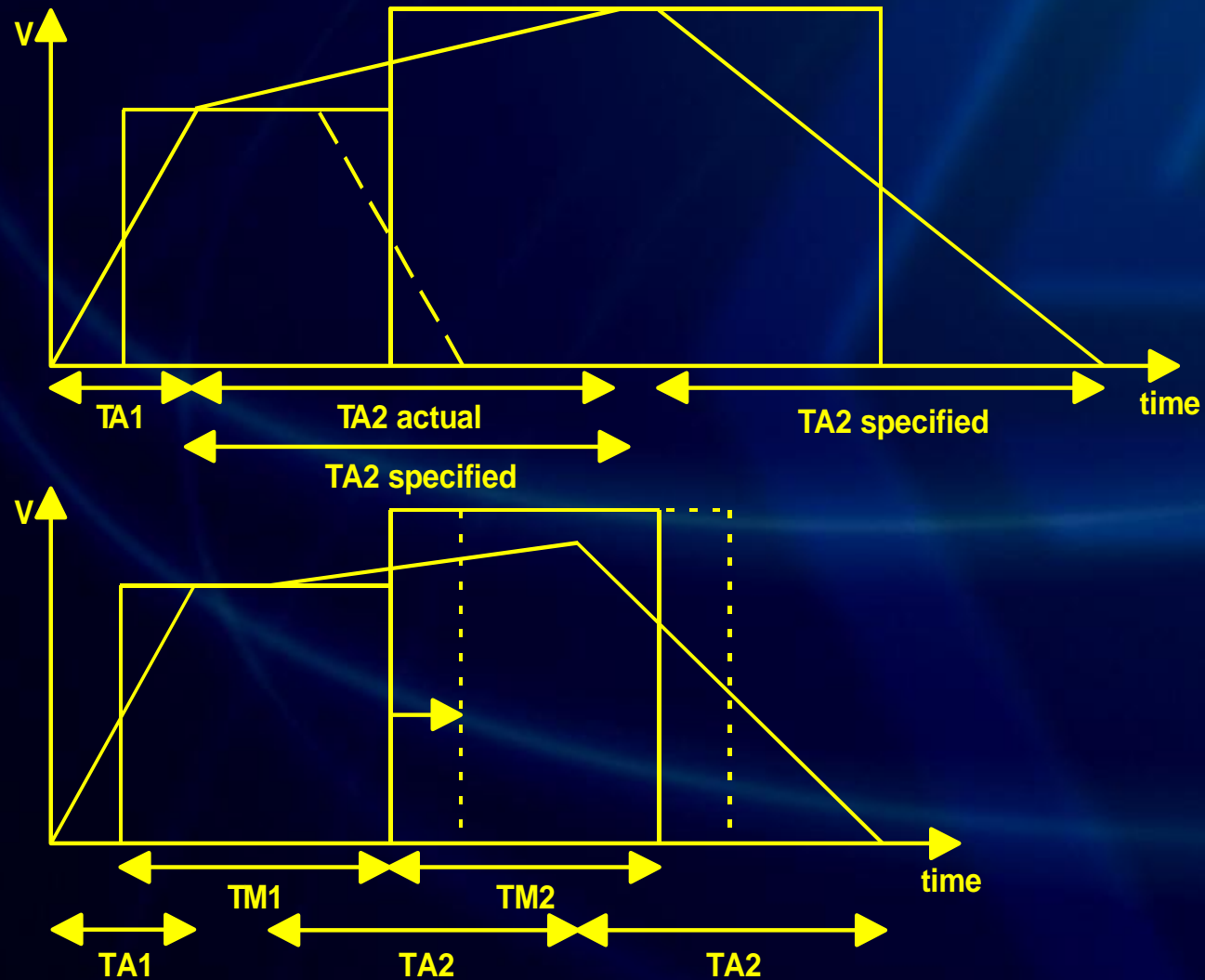
线性模式轨迹(续)

n 改变加速时间

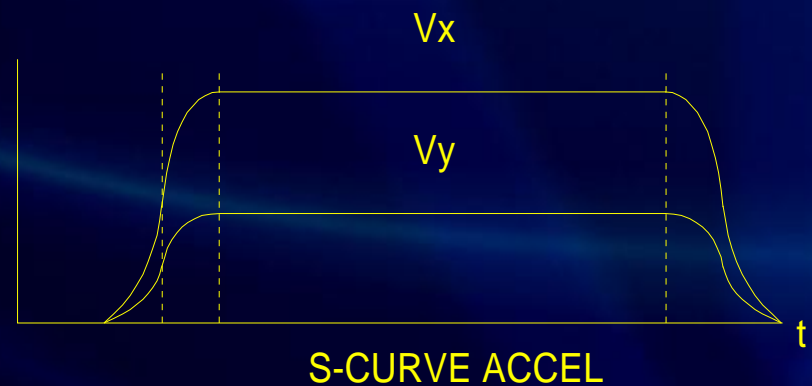
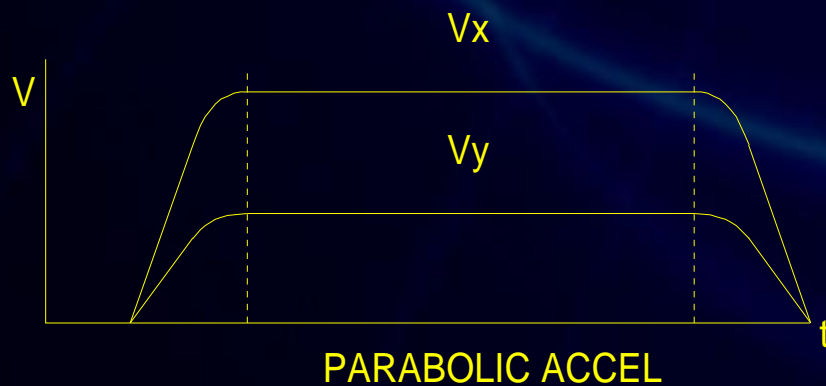
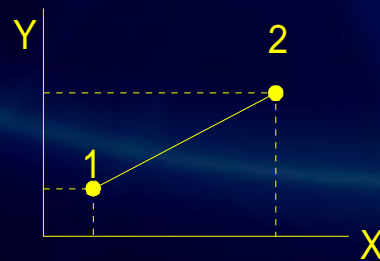
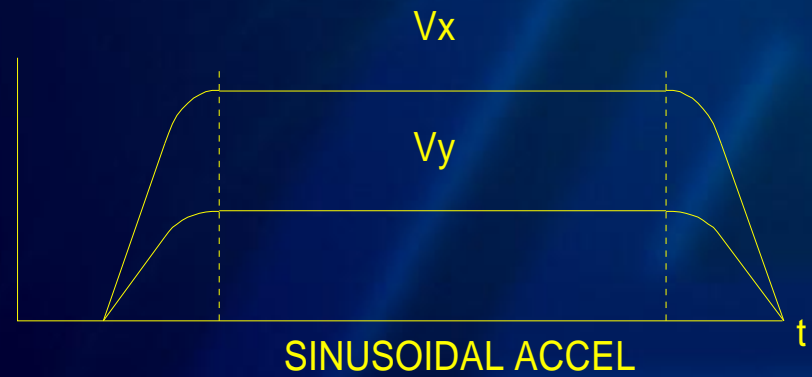
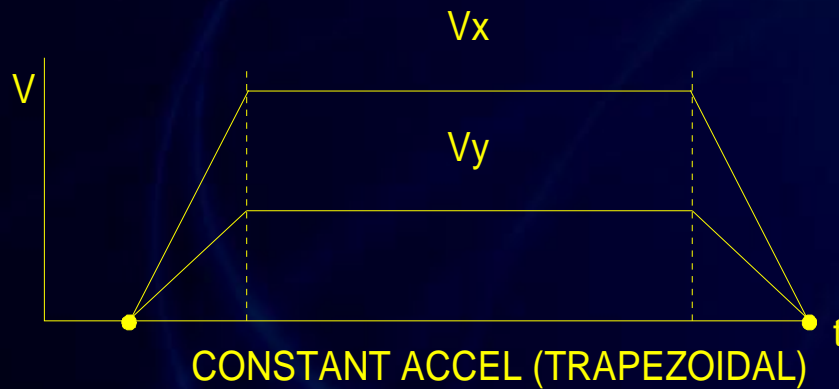


线性模式轨迹(续)

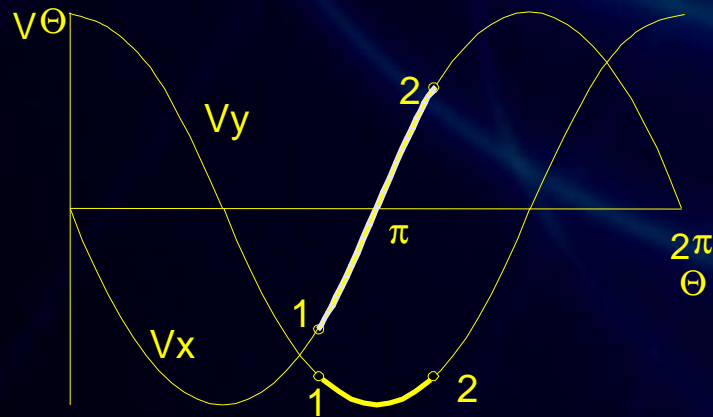
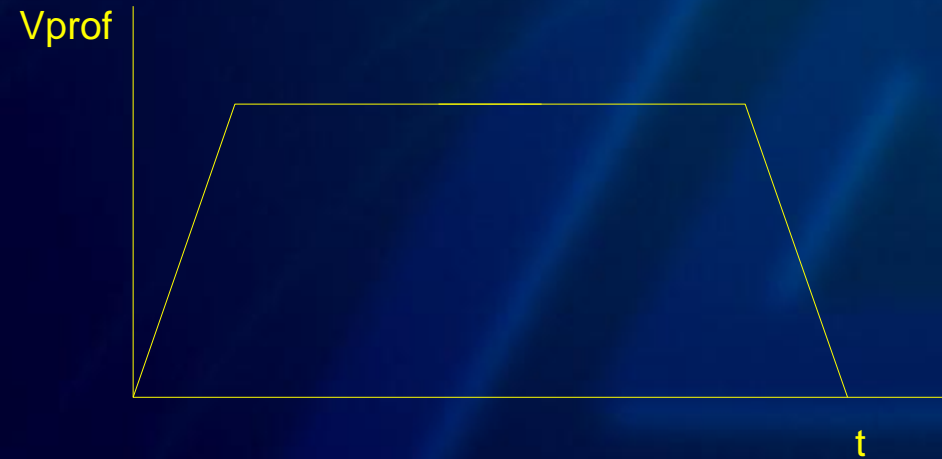
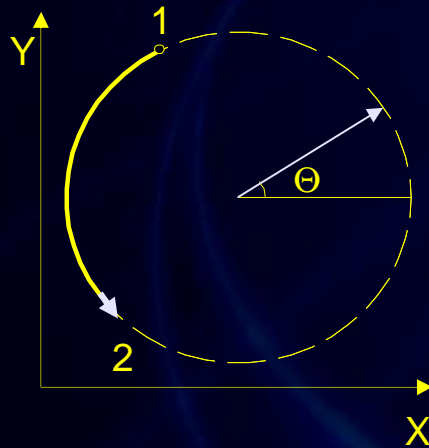
n 改变加速时间



曲线加速式线性插补

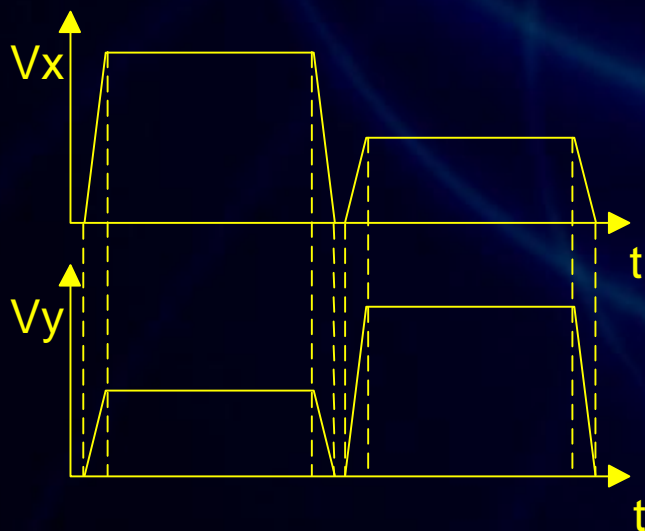
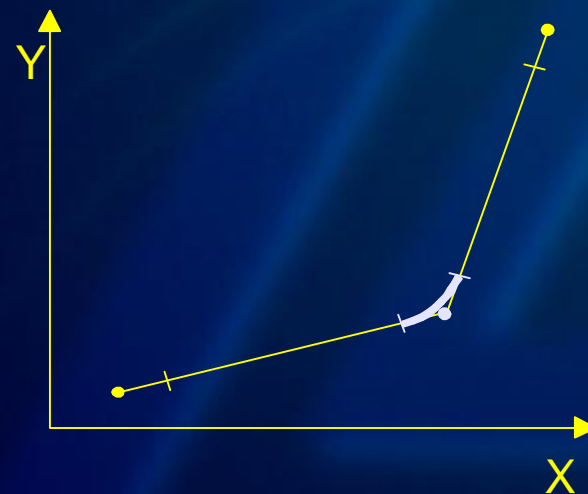
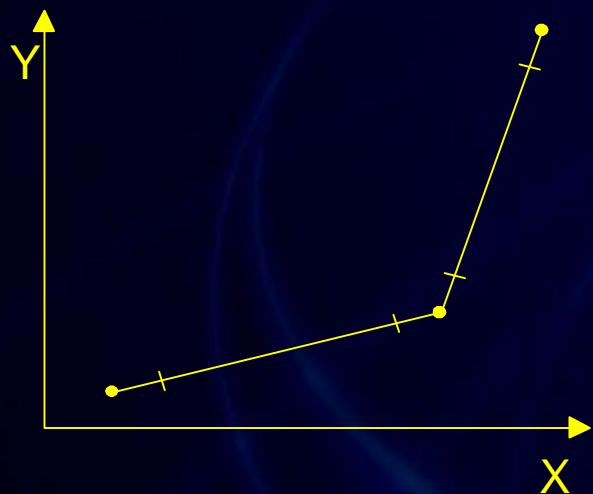


圆弧插补

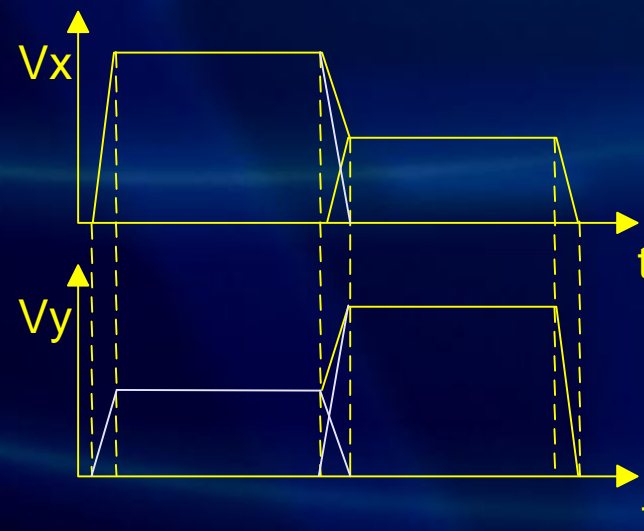


$$\begin{aligned} V_x(t) &= V_x(\Theta) \cdot V_{prof}(t) \\ &= R(-\sin \Theta) \cdot V_{prof}(t) \\ V_y(t) &= V_y(\Theta) \cdot V_{prof}(t) \\ &= R \cos \Theta \cdot V_{prof}(t) \end{aligned}$$

多轴线性运动 无混合与混合比较



Non-Blended



Blended

矢量进给率命令

n Frax (X,Y,Z):

- n 这个命令指定矢量进给率 (速率) 计算中所使用的轴
- n PMAC将所有进给率轴要运动的矢量距离(各相关轴要运动距离的平方和的平方根)除以进给率的值作为运动时间。
- n 在同一行内得到指令的那些非进给率轴也将在相同的时间内完成运动。

- n 用于多个电机控制一个工具时，各电机必须协调地工作获取期望的工具端速度。

矢量进给率轴

INC

FRAX (X,Y)

X3 Y4 F10

$$\text{距离} = \sqrt{3^2 + 4^2} = 5$$

$$\text{运动时间} = 5 / 10 = 0.5$$

$$V_x = 3 / 0.5 = 6$$

$$V_y = 4 / 0.5 = 8$$

INC

FRAX (X,Y)

X3 Y4 Z12 F10

$$\text{距离} = \sqrt{3^2 + 4^2} = 5$$

$$\text{运动时间} = 5 / 10 = 0.5$$

$$V_x = 3 / 0.5 = 6$$

$$V_y = 4 / 0.5 = 8$$

$$V_z = 12 / 0.5 = 24$$

矢量进给轴 (续)

INC
FRAX (X,Y,Z)
X3 Y4 Z12 F10

$$\text{距离} = \sqrt{3^2 + 4^2 + 12^2} = 13$$

$$\text{运动时间} = 13 / 10 = 1.3$$

$$V_x = 3 / 1.3 = 2.31$$

$$V_y = 4 / 1.3 = 3.08$$

$$V_z = 12 / 1.3 = 9.23$$

INC
FRAX (X,Y)
C10 F10

$$\text{距离} = 0$$

$$\text{运动时间} = 0 / 10 = 0 (< T_A)$$

加速度 - 限制运动

进给时间单位

$$F \text{ 单位} = \frac{\text{在定义电机时所定义的用户单位}}{\text{每 } Isx90}$$

$$Isx90 \text{ 进给率时间单位} \quad Fx = \frac{x \text{ 用户距离对应的计数}}{Isx90}$$

(单位: msec)

对于坐标系1-9, s=5, x为坐标系编号;
对于坐标系10-16, s=6, x为坐标系编号减10;

例子:

```
#1->2000X
Isx90=1000    ; 进给率单位: 1000毫秒
Open Prog 1 Clear
lin
inc
F5            ; 进给率 = 5 x 2000cnt/1000 msec
X3            ; 3 x 2000 counts
..
Close
```

进给率小测试

- n 若 $I5190=60000$ ，坐标系1的进给时间单位是什么？
- n 若X和Y都是进给轴，则运动X100 Y200的命令给进率中，是轴 X 还是轴Y将运动？
- n 如果编程命令F5，指定进给率为5英寸/秒，则必须如何设置Isx90，以及如何定义轴？

同步M-变量

同步M-变量赋值用来解决运动提前计算引起的同步问题。

当遇到同步M-变量赋值，并不立即执行；而是放入堆栈中，在程序的下一个运动开始时执行。因此输出动作与运动动作同步

<u>例子：</u>	LINEAR	;线性运动方式
	X10	;使X-轴运动到位置10
	X20	;使X-轴运动到位置20
	M1==1	;打开输出 #1
	X50	;使X-轴运动到位置50

在运动开始到X10，遇到语句M1==1，但是不进行真的赋值，直到混合到运动X50开始

PMAC 数据采集器

PMAC的内置式数据采集系统

PMAC 数据采集

PMAC的内置式数据采集：PMAC Plot Pro2

- n 任意PMAC地址的实时采样
- n 一次采集最多48个地址， 每个地址24或48位宽，（由变量I5001-I5048指定； 由变量I5050和I5051掩蔽）
- n 采样率为1到8百万个伺服周期（I5049）
- n 由采集数据产生的绘图和表格可以很容易的上传到主机，以方便处理和分析
- n 用于：系统辨识、伺服环整定、程序调试、机床调试、维护

采样命令

DELETE GATHER	; 清除数据采集缓存
DEFINE GATHER [{常数}]	; 设置数据采集缓存区, 所有未使用的 ; 内存都为数据采集保留 [或指定保留的大小]
GATHER [触发器]	; 开始数据采集 [外部触发]
ENDGATHER	; 停止数据采集
LISTGATHER	; 报告数据采集缓存内容

数据采集处理和采样率

数据采集可在终端窗口，运用运动程序的在线命令或者PLC程序进行初始化。

终端窗口中的在线命令：

DELETE GATHER 或 **DELGAT**
DEFINE GATHER 或 **DEFGAT**
GATHER 或 **GAT**
END GATHER 或 **ENDG**

释放内存分配
准备好PMAC以采集数据
开始PMAC的采集处理
结束PMAC的采集处理

在运动程序或PLC中：

COMMAND "DELETE GATHER"
COMMAND "DEFINE GATHER"
COMMAND "GATHER"
COMMAND "END GATHER"

等同于
等同于
等同于
等同于

CMD"DELGAT"
CMD"DEFGAT"
CMD"GAT"
CMD"ENDG"

数据采集器的采样率由**I5049**设置 (在伺服周期内).
$$I5049 = (\text{伺服周期频率}) / (\text{期望的采样率})$$

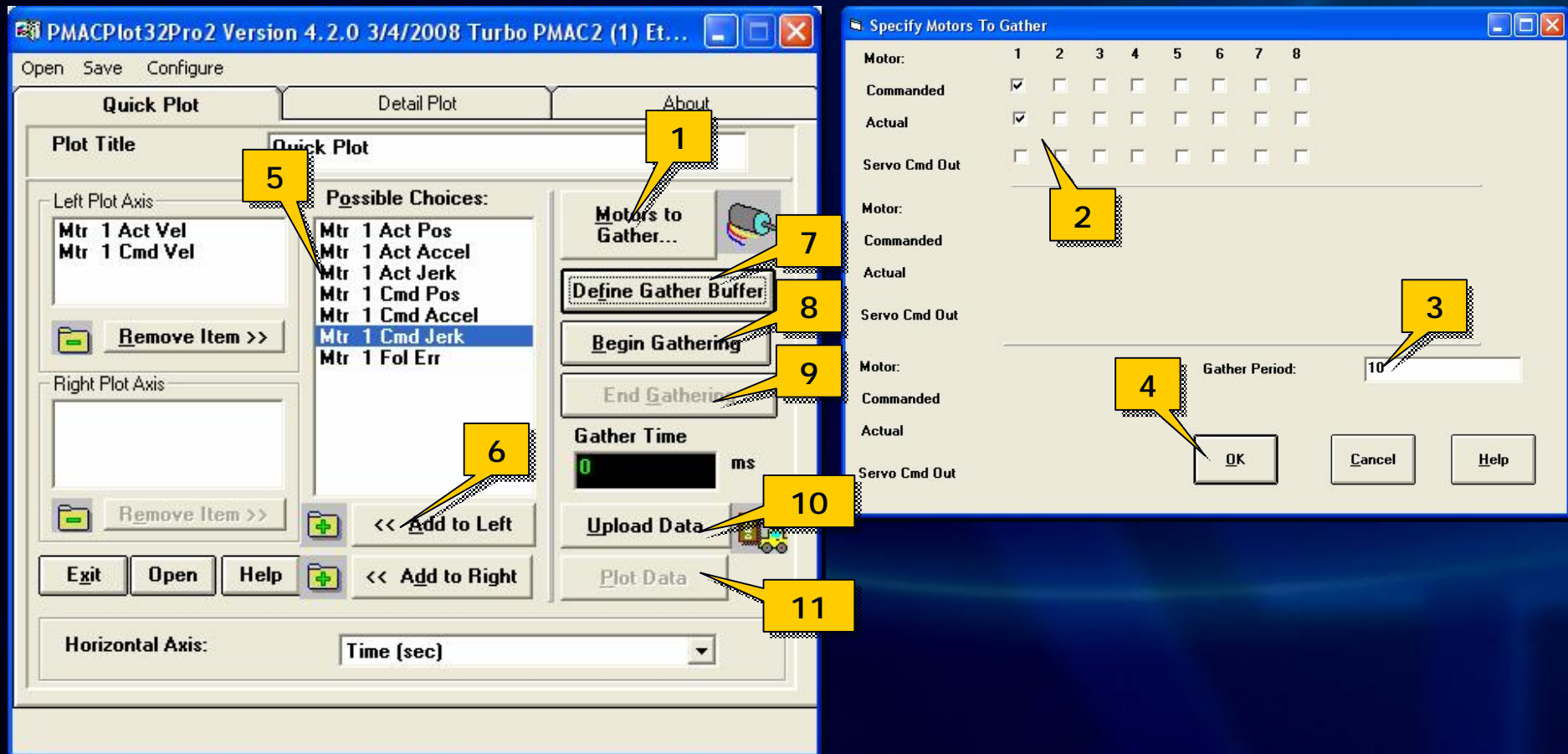
(I5049必须为整数)

PMAC Plot Pro 2

PMAC的执行PRO2软件包提供了PMAC Plot Pro2软件，它具有利用PMAC的数据采集能力的工具。

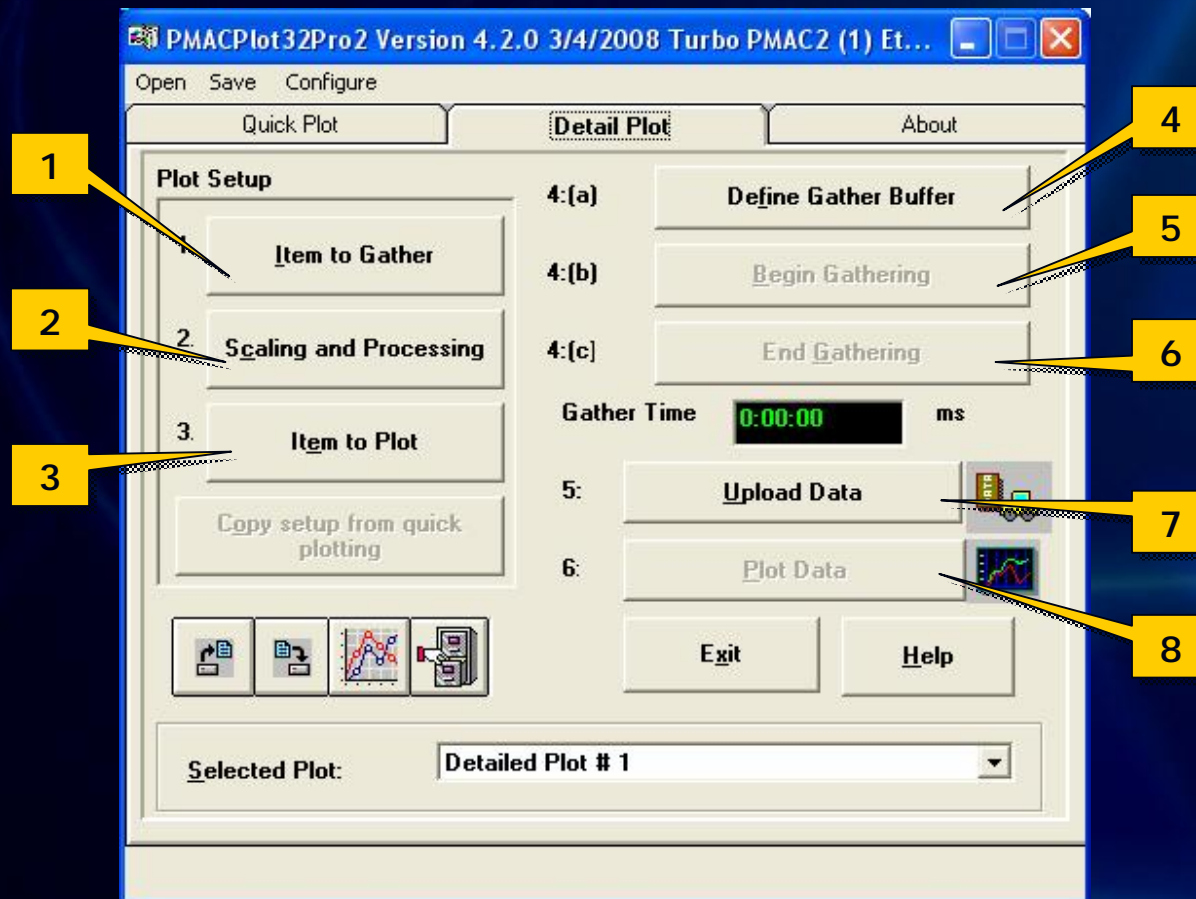


PMAC Plot Pro2 – 快速绘图



PMAC Plot Pro2 – 详细绘图

- n 详细绘图可以生成事实上来自任何PMAC地址数据的图形

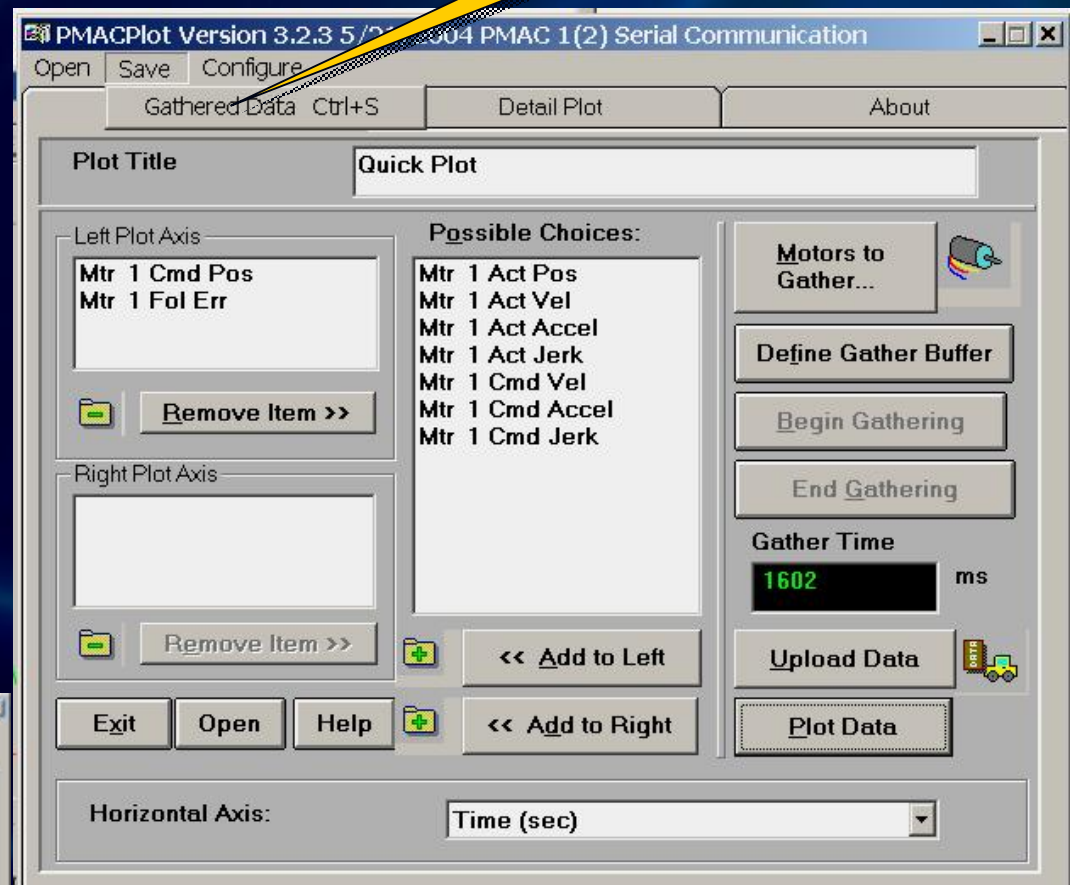
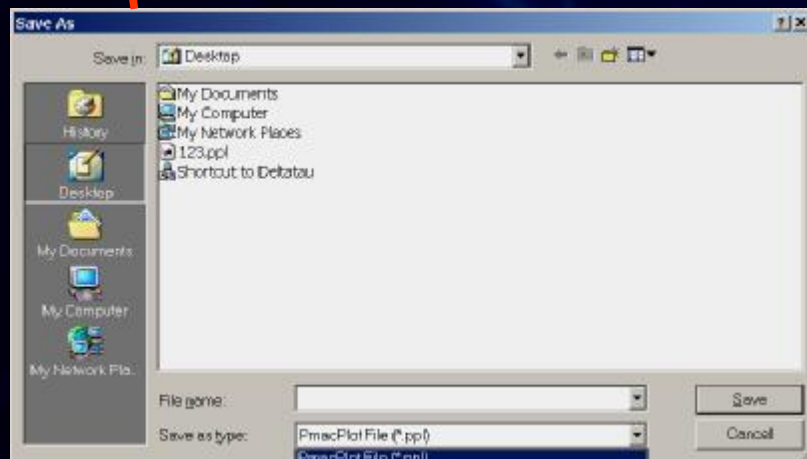
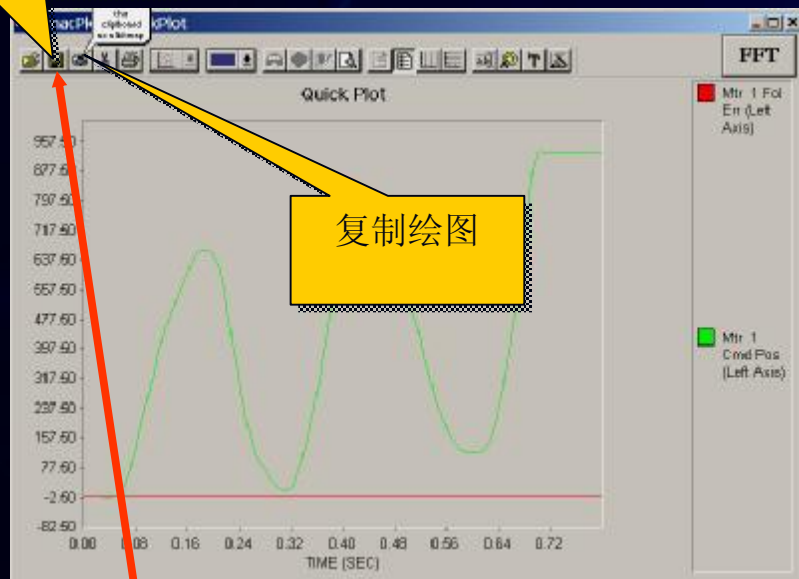


PMAC Plot Pro2 – 存储采集数据

保存绘图供
以后查看

复制绘图

保存
原始数据



带数据采集的简单运动

```
/****** 设置和定义 *****/
Undefine All
DEL GAT          ; 清除所有定义的采样缓存
&1              ; 寻址坐标系 1
CLOSE           ; 确保所有缓存关闭
#1->X           ; 电机#1分配到x-轴 - x轴的一个
                ; 编程单位就是一个编码计数

/****** 运动程序文本 *****/

OPEN PROG 1      ; 打开缓存, 编写程序#1输入程序
CLEAR           ; 清除缓存内容
LINEAR          ; 线性插补运动方式
ABS             ; 绝对方式 - 运动由位置规定
TA500           ; 设置1/2秒(500毫秒)加速时间
TS0            ; 设置无s-曲线加速时间
F5000           ; 设置5000单位(cts)/sec的进给速率(速度)
X10000          ; 使x-轴运动到位置10000
DWELL500        ; 在此停顿1/2秒(500毫秒)
X0             ; 使x-轴运动到位置0
DWELL0          ; Stop program 前瞻
CMD "ENDG"      ; Send 在线 命令 to 停止数据采集
CLOSE          ; 关闭缓存 - 结束程序
```

1. 运行此程序和采集数据:

```
DEL GAT<CR>
DEF GAT<CR>
GAT &1 B1 R      ; 采样, 坐标系1指向程序1的开始, 运行
```

2. 运动程序中包含“定义”和“采集”命令, 因此你只需发送b1r命令, 运行运动程序并采集数据!

带数据采集的简单运动

```
/****** 设置和定义 *****/
Undefine All
DEL GAT          ; 清除所有定义的采样缓存
&1              ; 坐标系1
CLOSE           ; 确保所有缓冲关闭
#1->X           ; 将电机1分配到x-轴 - x轴的一个编程单位
                ; 就是电机#1的一个编码计数

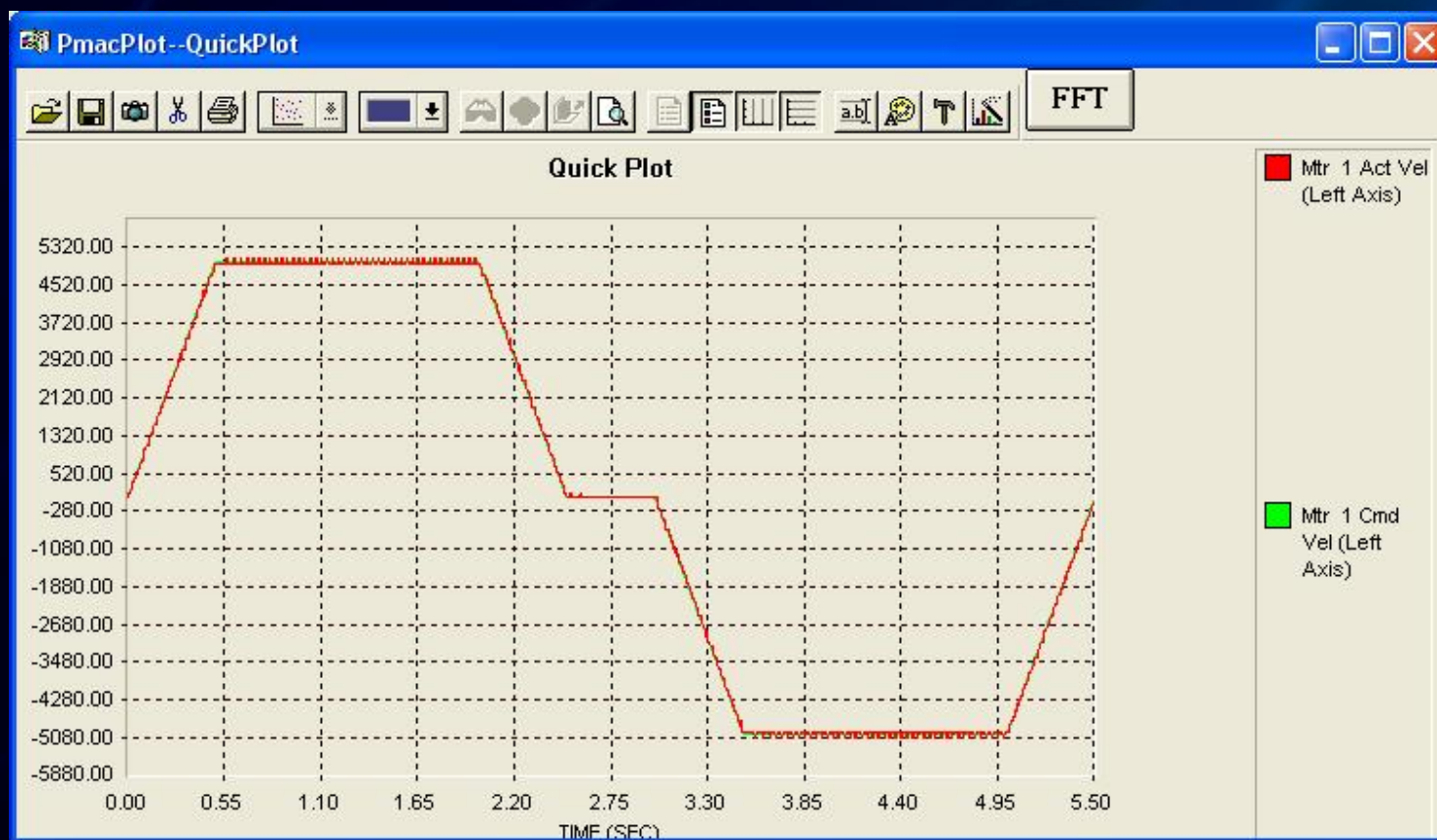
/****** 运动程序文本 *****/

OPEN PROG 1      ; 打开程序#1的缓存入口
CLEAR           ; 清除缓存内容
LINEAR          ; 直线插补运动方式
ABS            ; 绝对方式-运动由位置规定
TA500           ; 设置1/2秒(500毫秒)加速时间
TS0            ; 设置无s-曲线加速时间
F5000          ; 设置5000单位(cts)/sec的进给速率(速度)
CMD"DELGAT"      ; 删除已有采样缓存
CMD"DEFGAT"      ; 将所有可用的内存分配为新的采样缓存
CMD"GAT"        ; 开始数据采集
X10000         ; 使x-轴运动到位置10000
DWELL500        ; 在此位置停顿1/2秒(500毫秒)
X0             ; 使x-轴运动到位置0
DWELL0          ; 停止前瞻功能
CMD"ENDG"       ; 发送在线命令, 停止数据采集
CLOSE          ; 关闭缓存-程序结束
```

运行程序, 采集数据如下:

```
&1 B1 R          ; 坐标系1, 指向程序1的开始, 运行
```


带数据采集的简单运动



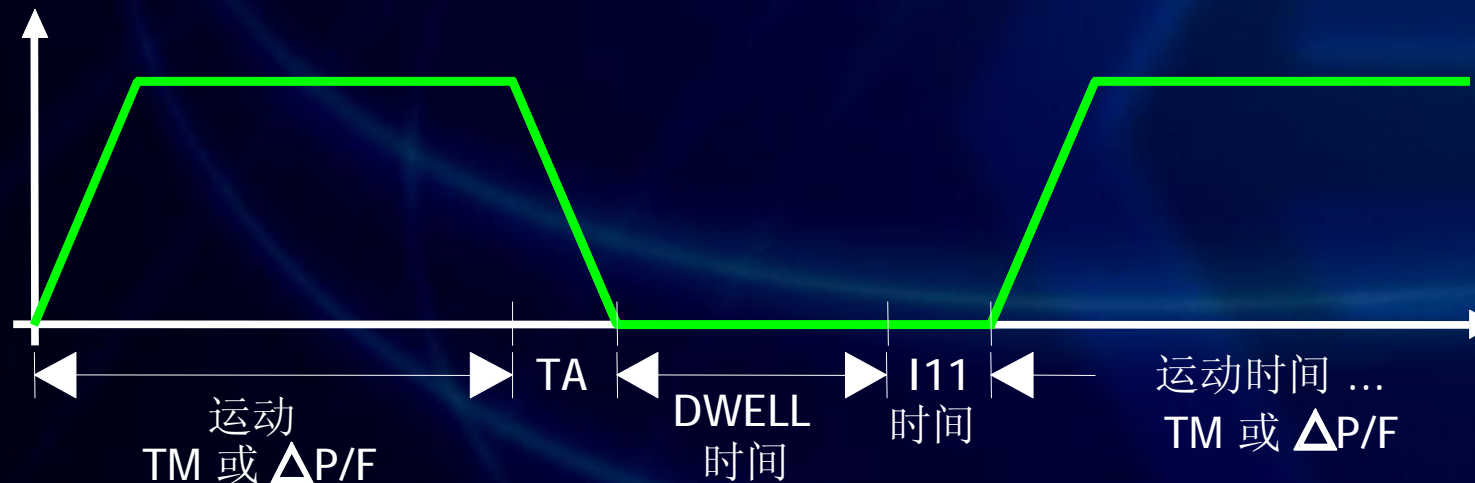
数据采集小测试

- n PMAC能采集什么信息数据?
- n 如果想数据采集速率为10 Hz (每秒10次), 则在伺服更新率为2.25 KHz的情况下, I19应该设置为多少?

DWELL Vs. DELAY

DWELL

- n 总是采用定时基 (I10)
- n 时间不包括前面的减速
- n 不对之后的运动做预计算，直到DWELL结束 (加上I11的时间)



I11: 程序运动计算时间 (默认值 = 0)

DWELL Vs. DELAY (续)

DELAY

- n 采样变时基 (% 值)
- n 时间包括前面的减速时间
- n 最小时间为电流TA时间
- n 接下来的运动计算在DELAY的开始时候就开始计算了



较复杂的运动

本例介绍增量运动和指定时间的运动，循环逻辑，变量应用，轴比例因子，以及简单的数学计算。逻辑和数学计算不会延迟任何运动。

```
*****设置和定义 *****
;
DEL GAT           ; 清除所有定义的采样缓存
&1               ; 坐标系1
CLOSE            ; 确保所有缓存关闭
#1->1000X         ; 电机#1的X轴的一个单位(cm)计数是1000
```

```
***** 运动程序文本 *****
```

```
OPEN PROG 2 CLEAR      ; 打开程序#2的缓存入口
                        ; 并清除缓存内容
LINEAR                ; 直线插补运动方式
INC                   ; 增量方式 - 运动量表示距离
TA500                 ; 1/2秒(500毫秒)加速时间
TS250                 ; 每半个S-曲线为1/4秒
TM2000                ; 2秒运动时间(减速开始)
P1=0                  ; 初始化循环计数变量
WHILE (P1<10)         ; 循环直到条件为假(10次)
    X10                ; X-轴正向运动10cm(=10,000 cts)
    DWELL500           ; 停止1/2秒
    X-10               ; X-轴反向运行10cm
    DWELL500           ; 停止1/2秒
    P1=P1+1            ; 循环计数器加1
ENDWHILE              ; 循环结束
CLOSE                 ; 关闭缓存 - 结束程序
```

运行此程序:

```
&1 B2 R              ; 坐标系2, 指向程序2开始, 运行
```

较复杂的运动(续)



轨迹小测试

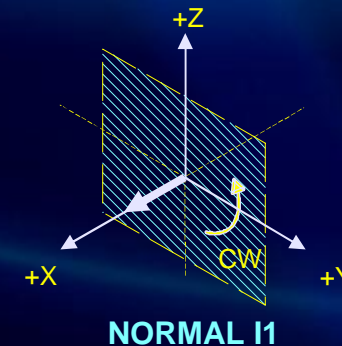
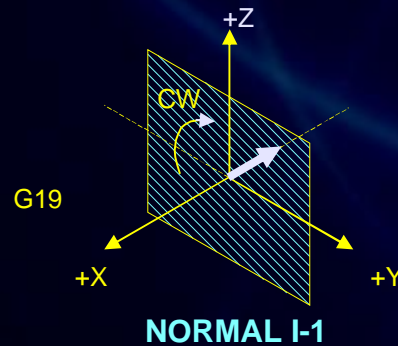
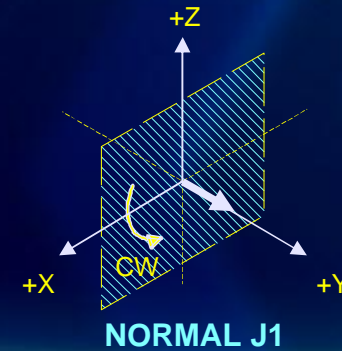
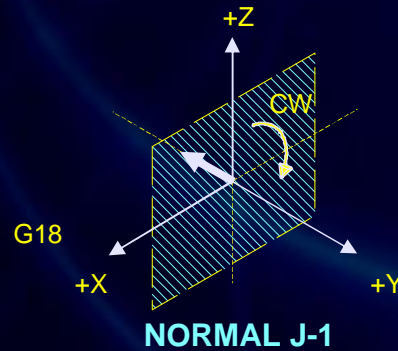
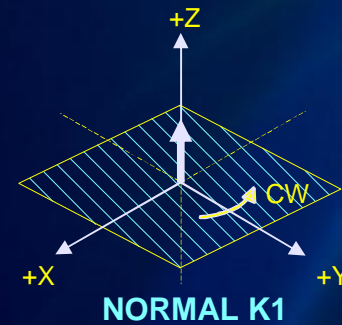
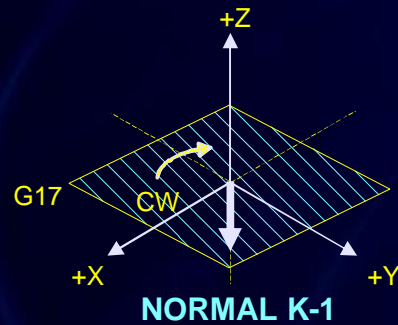
- n 绘图的水平轴和垂直轴表示什么？
- n 什么是TS, TA, TM 和 F？
- n TM和F如何联系？
- n 一个运动：TA时间为100、TS时间为20、TM时间为500，则多久能完成这一运动？
- n 如果程序声明TS100 TA100 TM100，则多久能完成3个混合运动？

圆弧插补运动模式

CIRCLE模式运动

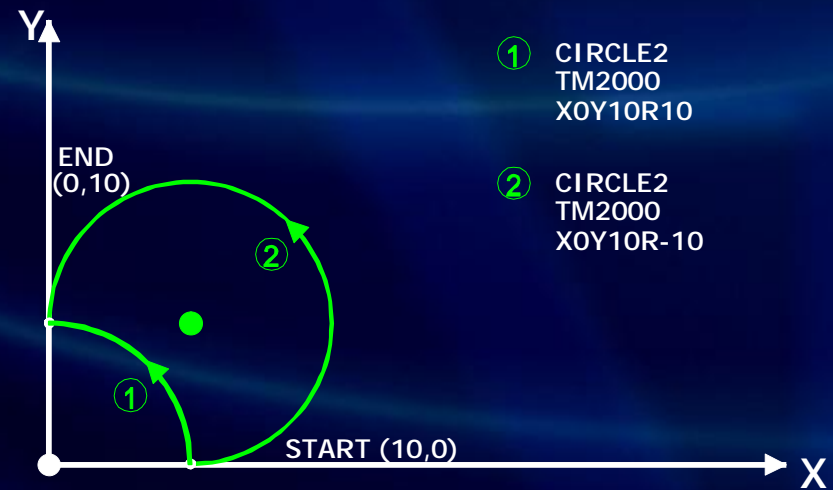
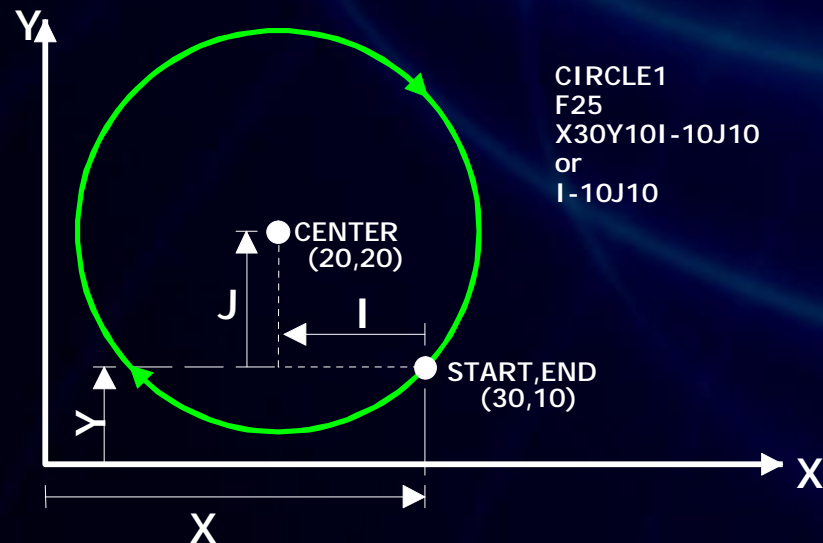
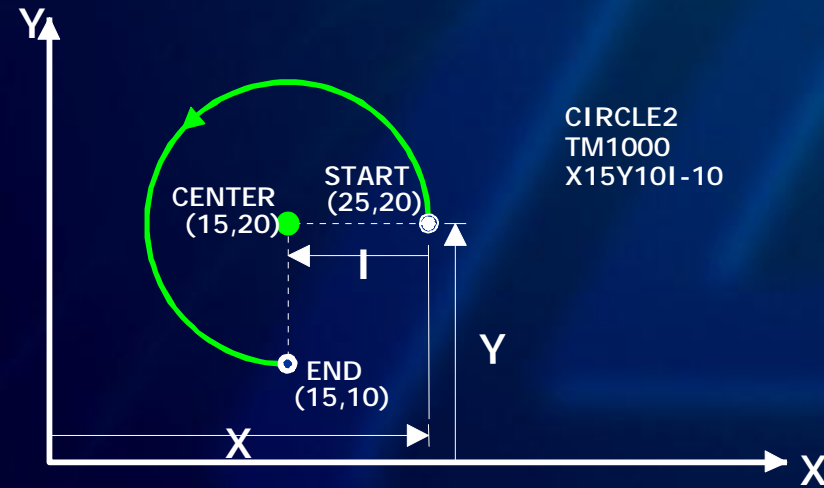
- n 圆面指定 (NORMAL)
- n 终点 (ABS)或距离(INC) 指定
- n 中心矢量(I,J,K)或半径(R) 参数
- n 矢量速度(F)或运动时间(TM)参数
- n 加速时间 (TA, TS)参数
- n 不在圆面的轴进行线性插补

圆弧插补：平面指定

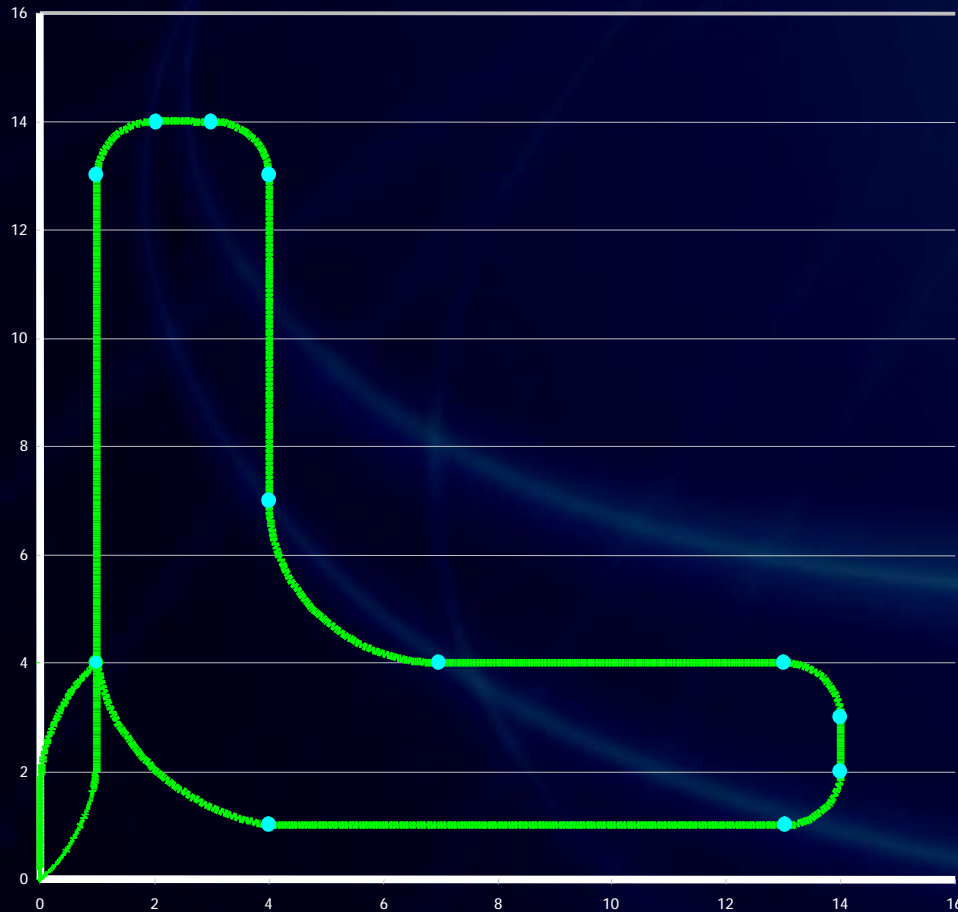


圆弧运动的法线矢量：
定义的平面和旋转方向

圆弧插补示例



圆弧插补示例(续)



;设置和定义

&1

#3->10000x

#4->10000y

i13=10

;运动程序文本

```
open prog4 clear
```

```
normal K-1
```

```
rapid x1 y4
```

```
f5
```

```
linear y13
```

```
circle1 x2 y14 i1 j0
```

```
linear x3
```

```
circle1 x4 y13 i0 j-1
```

```
linear y7
```

```
circle2 x7 y4 i3 j0
```

```
linear x13
```

```
circle1 x14 y3 i0 j-1
```

```
linear y2
```

```
circle1 x13 y1 i-1 j0
```

```
linear x4
```

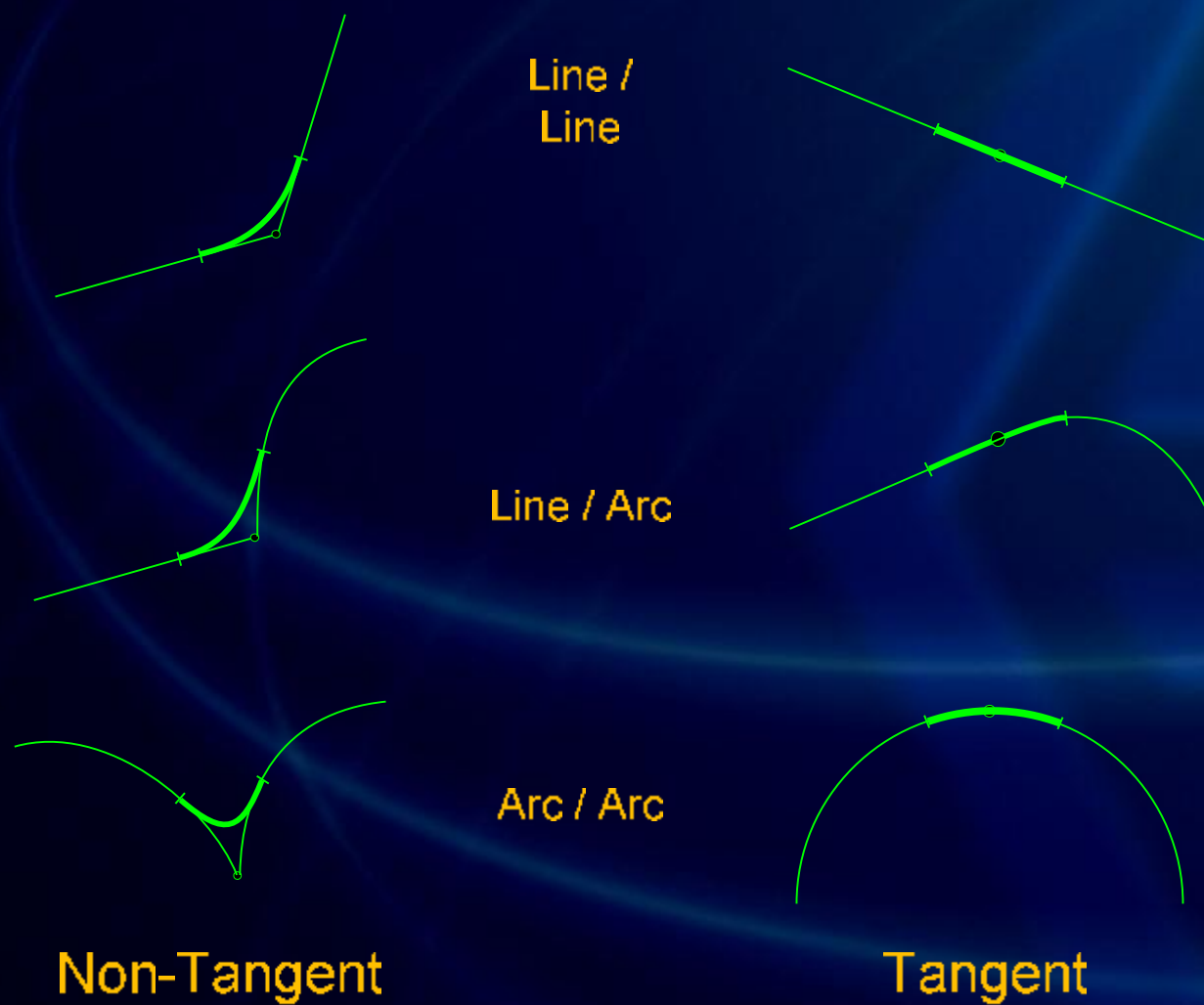
```
circle1 x1 y4 i0 j4
```

```
dwell1100
```

```
rapid x0 y0
```

```
close
```

直线和弧线之间混合



为什么使用分段模式 (Isx13>0)?

- n 允许圆弧插补 (Isx13是“匹配”时间)
- n 允许刀具半径 (Isx13是“匹配”时间)
- n 进给保持时允许手动 (“\”进给保持命令)
- n 允许执行运动结束时退出 (“/”命令)
- n 允许报告当前执行的运动命令
- n 旋转缓冲内，允许改变最终运动已计算出的减速度，以混合到新发送的运动指令。
- n 允许反向运动学 (Isx13是“匹配”时间)
- n 允许特别的前瞻和回溯分段缓冲区

PMAC的前瞻

什么是前瞻？

- n 所有Turbo PMAC的标准特性
- n 算法能在运动代码中进行“提前探查”
- n 检查是否超越了电机限制：
 - n 位置(软件超程)限制
 - n 速度限制
 - n 加速度限制
- n 当监测到超越，运动将在问题点缓行
- n 算法将在问题点之前重新计算运动，使所有的运动在不会超过限制

前瞻能做什么？

- n 前瞻允许多运动块(程序指令)之间加速和减速
- n 前瞻允许在急转弯前就开始减速，出来后再加速
- n 可以只需在急转弯处减速，而不需要在整个范围减慢速度，从而可以延长机械寿命
- n 前瞻允许在位置限制处停止，而不会超过限位
- n 前瞻不会改变程序路径，只是更快的运行完路径

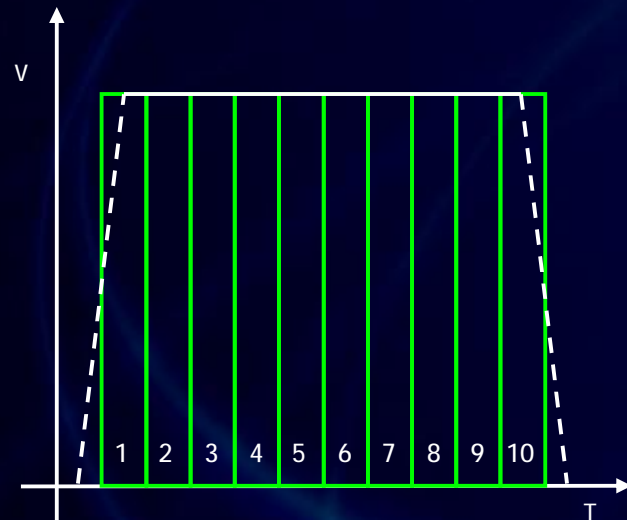
什么是“分段前瞻”？

- n 前瞻利用了PMAC的分段 (粗插补) 功能
- n 前瞻缓冲存储相对一致的运动段，不是无法确定的程序运动块

运动块来自编程的原始指令，一个运动块就是一组坐标轴指令，是轨迹的原始描叙；运动段是PMAC用初插在轨迹上进行分段

- n 分段的一致性允许：
 - n 更有效的计算
 - n 对长短不同的运动块有更好的优化解决方案
- n 紧致的分段使得：
 - n 加速和减速在多个程序块进行，不是在每个块内都要进行独立的加减速
 - n 圆弧运动可以线性逼近
- n 分段是可逆的；运动程序不可逆！

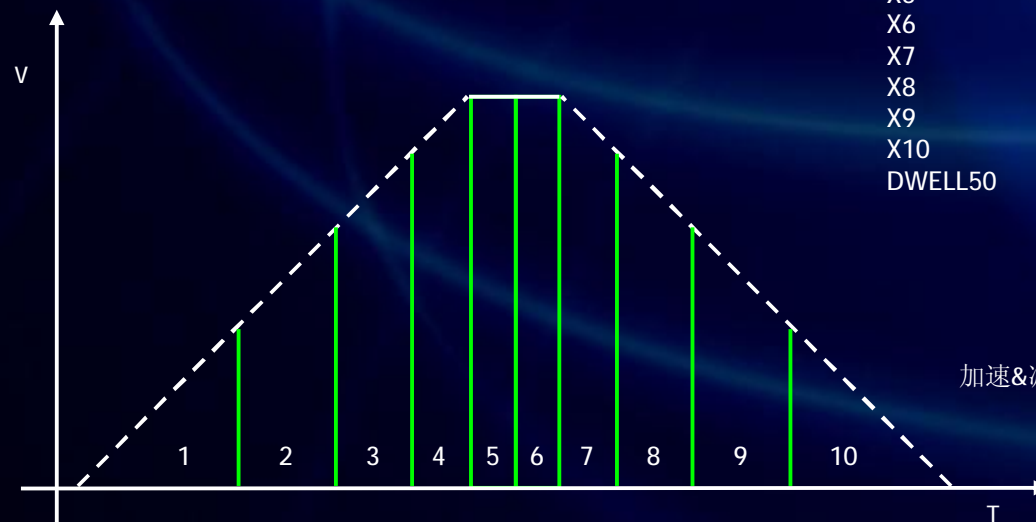
多块加速/减速的前瞻



加速&减速时间必须等于或小于块运动时间

前瞻之前

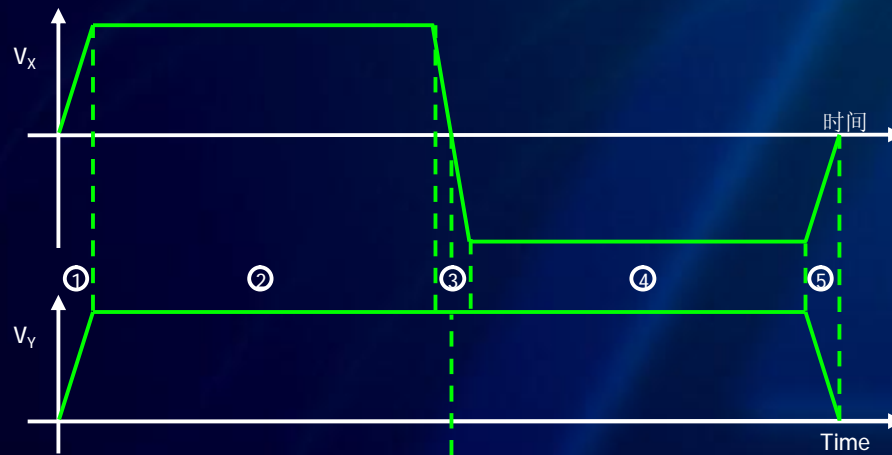
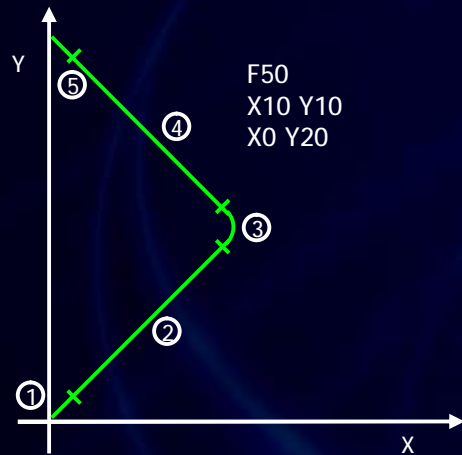
F100
X1
X2
X3
X4
X5
X6
X7
X8
X9
X10
DWELL50



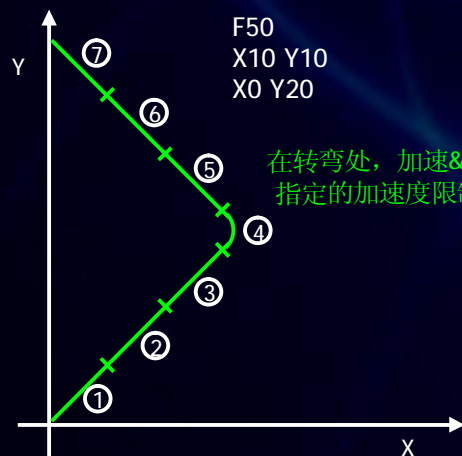
前瞻之后

加速&减速被指定的加速限制控制 (G's)

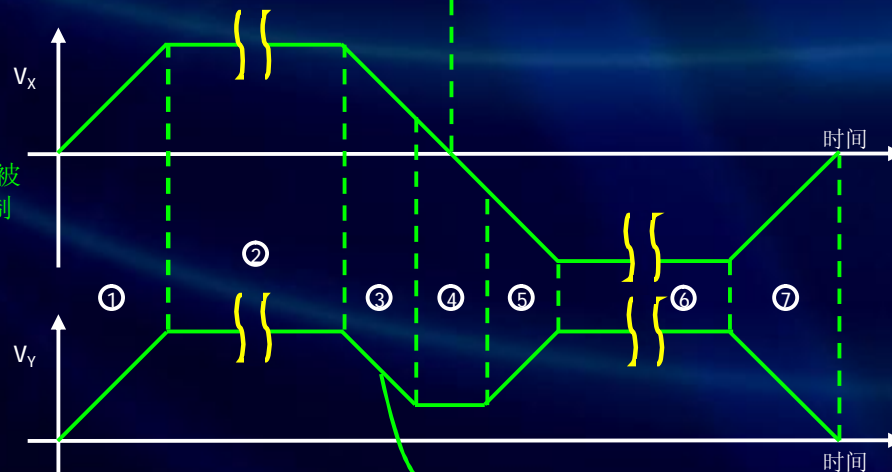
前瞻 & 小的急转弯



前瞻之前



在转弯处，加速&减速被指定的加速度限制控制



前瞻之后

前瞻的关键变量

- n lxx13 电机 xx 正位置限位
- n lxx14 电机 xx 负位置限位
- n lxx16 电机 xx 最大速度
- n lxx17 电机 xx 最大加速度
- n lxx13 C.S. 'x' 段时间
- n DEFINE LOOKAHEAD {# of 段}, {# of 输出}
- n lxx20 C.S. 'x' 前瞻距离 (分段数)

需要多少前瞻?

- n 需要提前知道停止的距离/时间
- n 最坏的情况下，停止时间为最大速度 (v_{max}) 除以最大加速度 (a_{max})
- n 为所有的电机选择最大时间
- n 提前需要的段数为：
停止时间 / (2 * 段时间)
- n 让PMAC 前瞻需要段数 4/3倍，这样会有足够的余量
- n 在此基础上设置前瞻缓冲区，必要时还要考虑回溯需要的段数

Turbo 的前瞻速度能力

- n 2轴: 2000块/秒 (1 msec seg, 200%)
- n 3轴: 1000块/秒 (1 msec seg)
- n 4 轴: 500+块/秒 (2 msec seg)
- n 5 轴: 500+块/秒 (2 msec seg)
- n 6 轴: 500块/秒 (2 msec seg)
- n 8 轴: 333块/秒 (3 msec seg)
- n 12 轴: 250块/秒 (4 msec seg)
- n 16 轴: 200块/秒 (5 msec seg)

分段时间 = 最小块时间;

缺省的伺服更新率为2.25 kHz, 无换相

80MHz CPU

Turbo 前瞻长度能力

- n 前瞻的长度仅被内存能力限制！
- n 更长的前瞻不需要更快的处理速度！
- n 标准的数据内存, 90% 用于前瞻:
 - n 2轴: 3000 段
 - n 4轴: 2000 段
 - n 6轴: 1500 段
- n 扩展的数据内存, 90% 用于前瞻:
 - n 2轴: 24,000 段
 - n 4轴: 16,000 段
 - n 6轴: 12,000 段

回溯前瞻

- n 超容的前瞻缓冲可以提供轨迹的“历史”
- n 缓冲中的位置信息是可逆的（不像实际运动程序）
- n ‘<’ 命令可使沿路径逆转
- n 逆向操作遵守前向操作相同的限制
- n 逆向长度仅受缓冲内存的限制
- n 只能在连续混合序列内逆向
- n 用‘\’ 命令可在任何一个方向快速停止
- n ‘>’, ‘R’, 或 ‘S’ 命令再次前向操作
- n 无缝连续到程序新的部分

Non-Turbo PMAC上的前瞻

- n Option 6L是前瞻固件
- n 与标准固件轻微的不兼容
- n 前瞻限制在1个坐标系
- n 比在Turbo上更低的速度
- n 在前瞻时间上无位置限制
- n 任何PMAC或PMAC2版本都具有此功能
- n 最重要的是在低价的Mini和Lite 控制器上具有此功能

快速模式运动

Rapid 模式运动

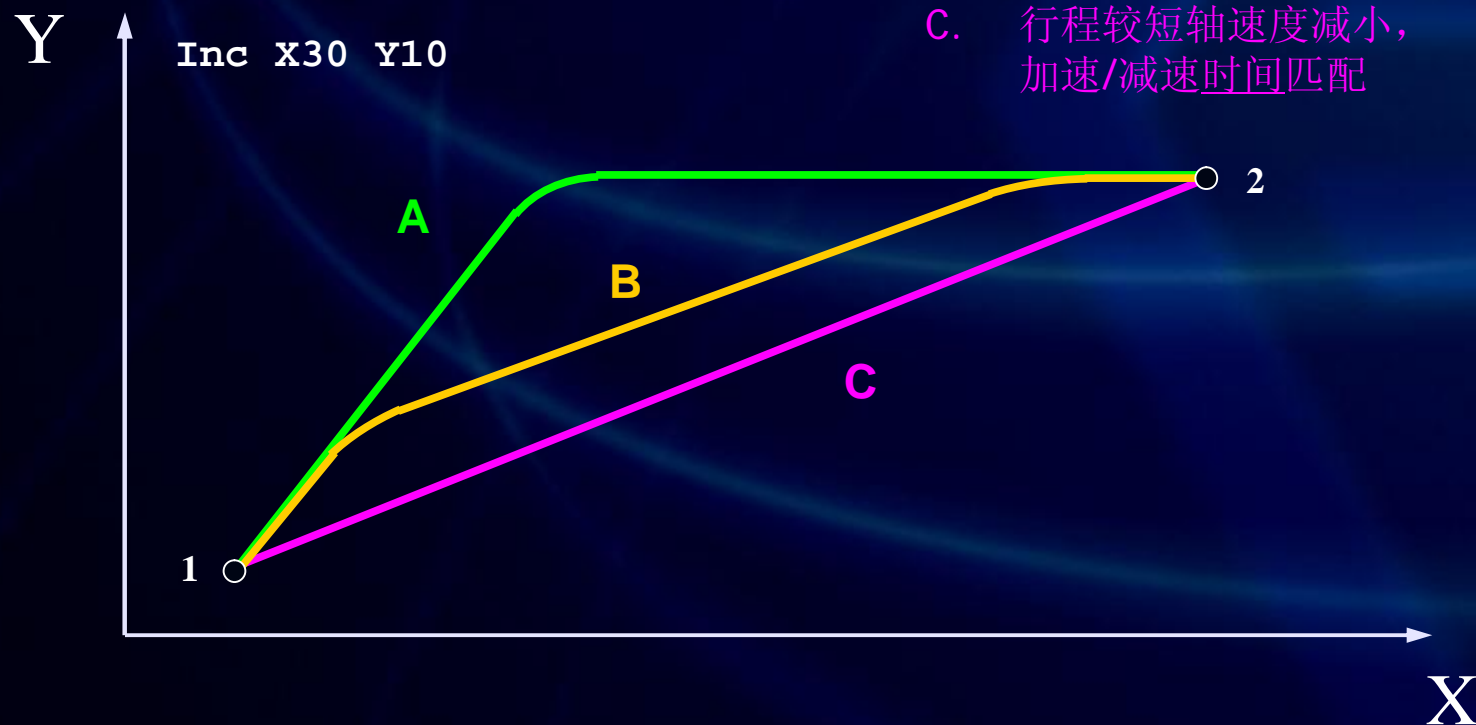
- n 终点 (ABS)或距离(INC)指定
- n 最大速度(lxx16) 或手动速度(lxx22) 参数
- n 加速时间 (lxx20, lxx21) or 速率 (lxx19) 参数
- n 只有具有最高距离/速度轴(最长时间)以给定指定的速度运行；其它的轴匹配这个时间
- n 如果加速时间匹配的话走的是直线路径
- n 在不同的方向无恒矢量速度
- n 触发运动： 例. X100^2.5
- n 改变目的地： 例. ! X19.93 Y31.25 (仅Turbo)

Rapid模式路径选项

A. 行程较短轴全速运行
(Isx79=1, Turbo 固件 ≥ 1.941)

B. 行程较短轴速度减小,
加速/减速率匹配

C. 行程较短轴速度减小,
加速/减速时间匹配

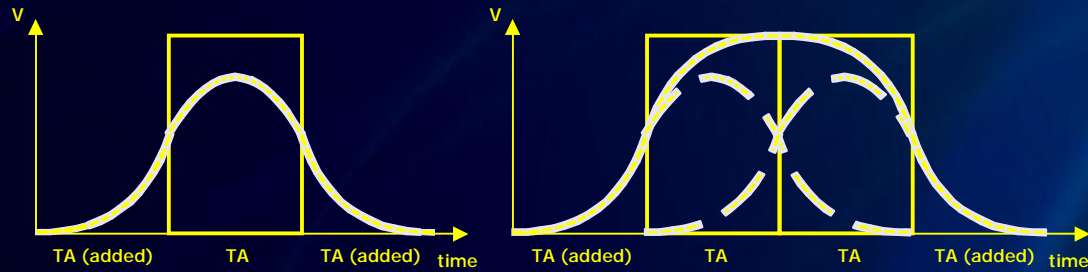


样条曲线模式运动

样条曲线模式运动

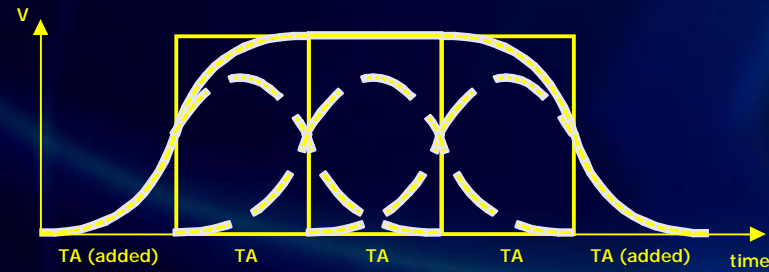
- n 终点 (ABS) 或 距离 (INC) 指定
- n Non-Turbo: 运动时间 (TA) 参数, 近毫秒级别
- n Turbo: 运动时间 (TM) 规范, $\frac{1}{4}$ 微秒分辨率
- n SPLINE1 模式要求定长运动时间
- n SPLINE2 模式允许变运动时间, 但只应该慢变
- n 位置、速度和加速度在运动块边界上连续
- n 混合近似到编程点内部
- n 产生 立方 B-样条路径
- n PMAC模式中最高效的计算
- n 内部用作分段插补(初插)

Cubic 样条轨迹

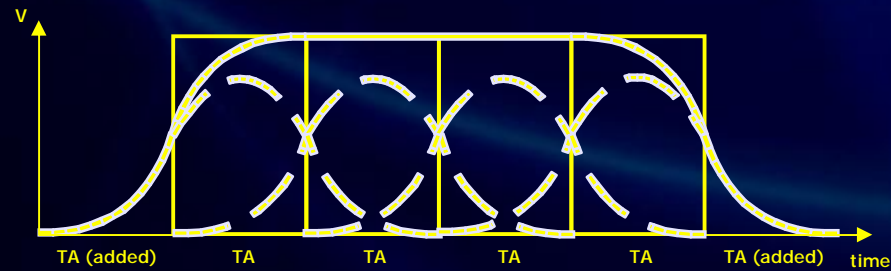


一个程序段

两个程序段



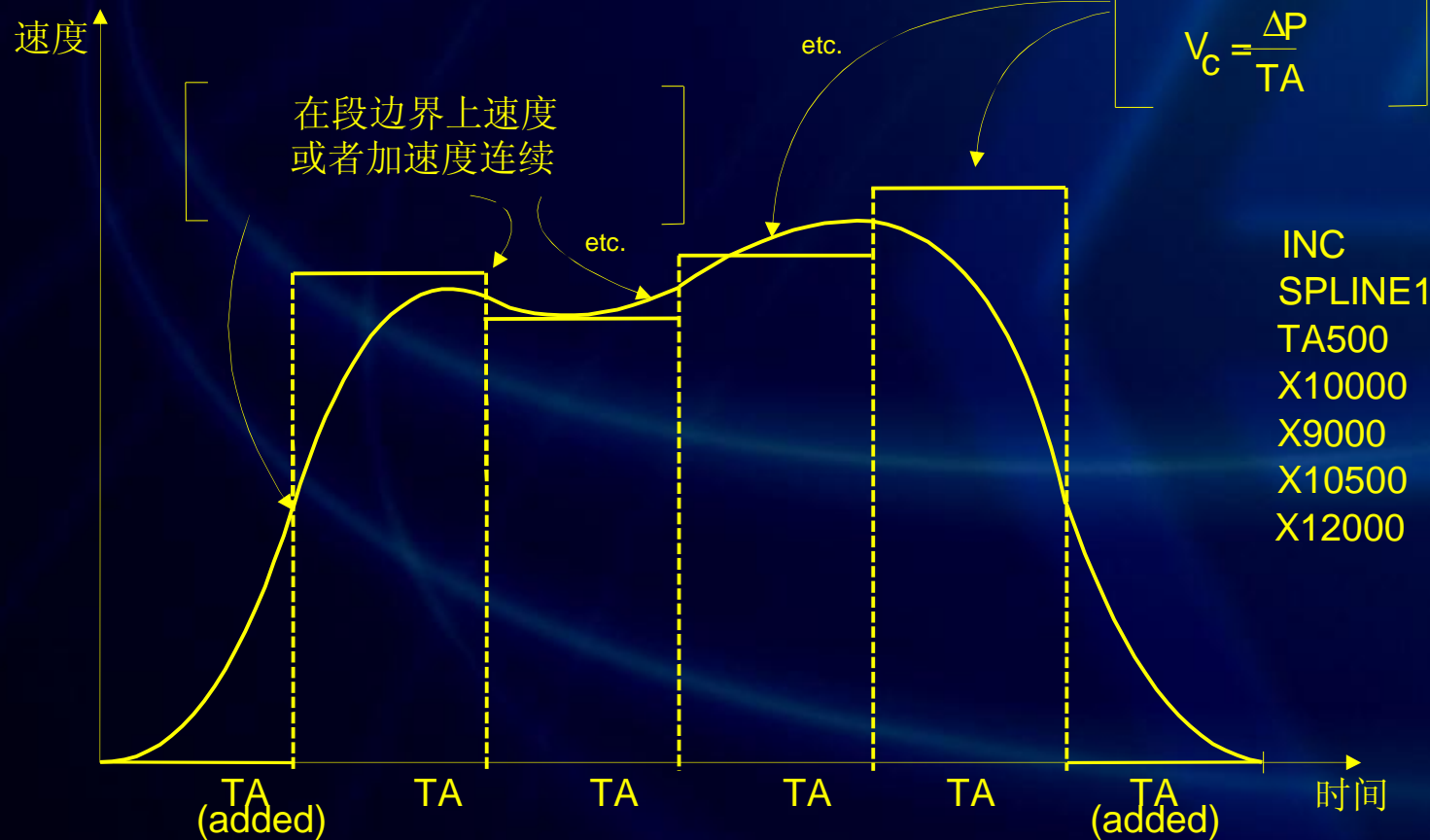
三个程序段



四个程序段

PMAC的样条曲线运动

所有分段具有相同的时间



PVT 模式运动

位置、速度、时间

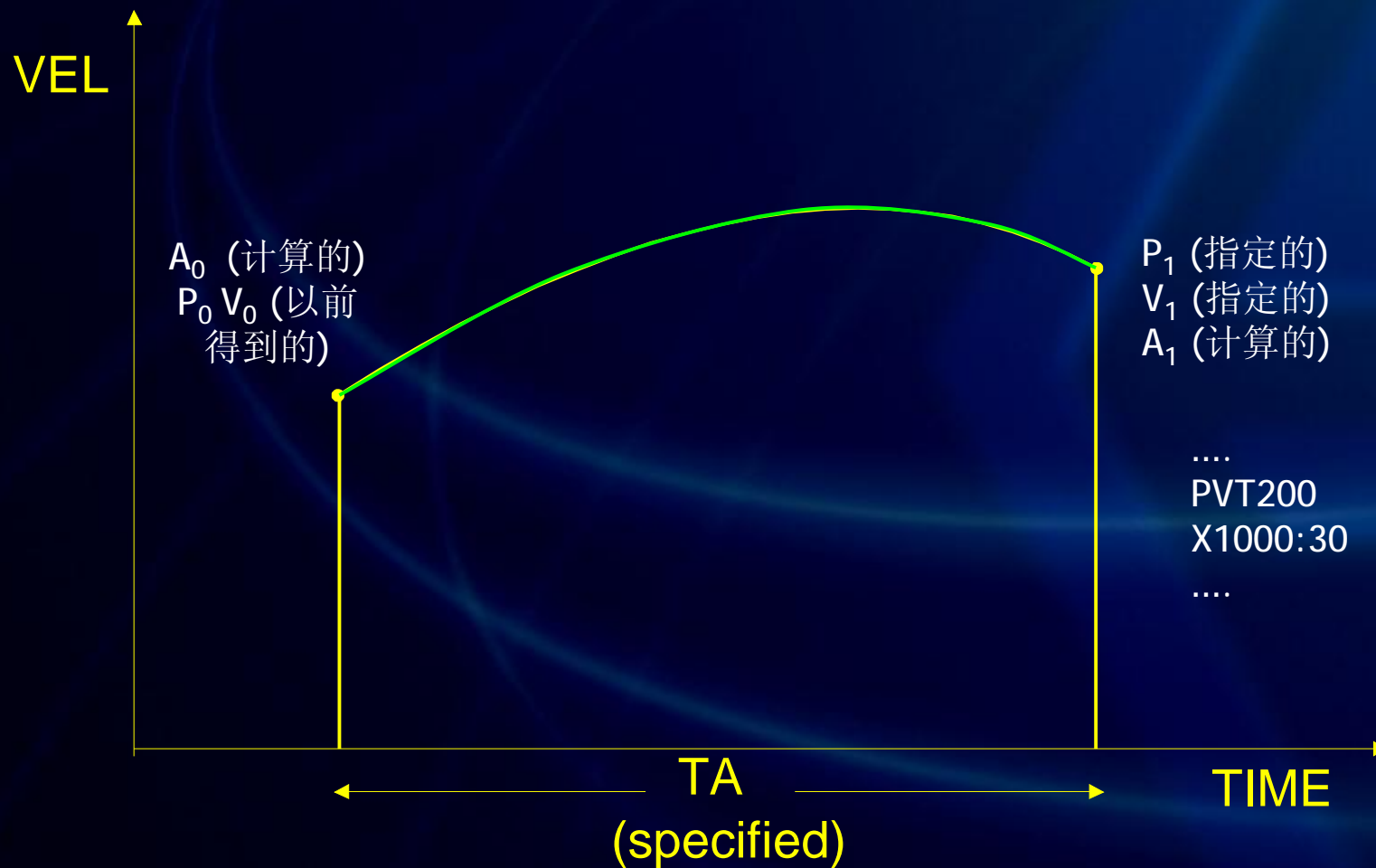
PVT 模式运动

- n 终点(ABS) 或 距离 (INC) 指定
- n 对于运动块，每个轴上要指定有符号的速度
- n [Non-Turbo]: 采用运动时间 (TA) 来指定, 毫秒精度
- n [Turbo]: 采用运动时间 (TM) 来定义, 1/4-微妙精度
- n 在运动边界上，位置和速度连续

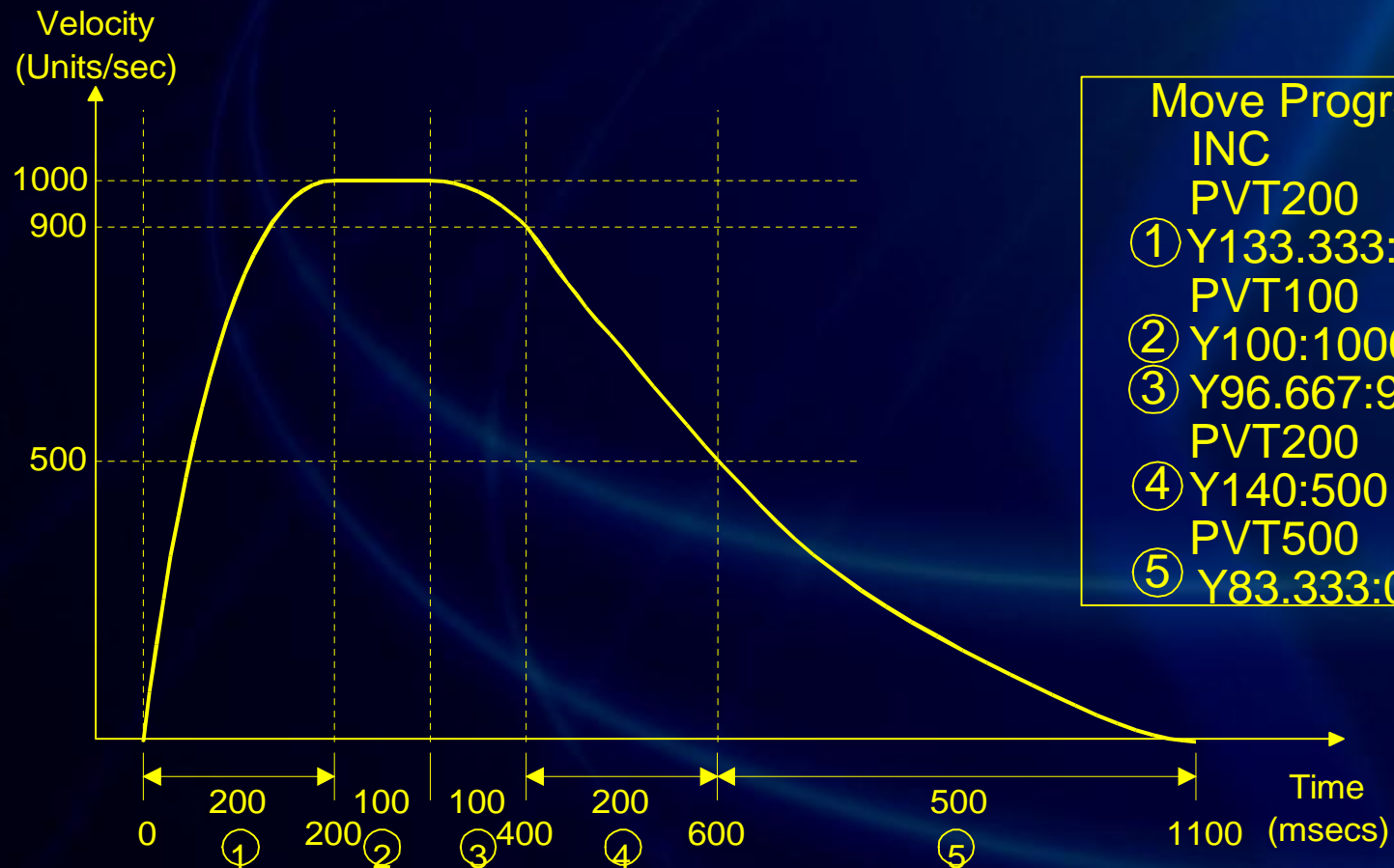
PVT 模式运动(续)

- n 混合会准确的经过程序点
- n 利于创建任意曲线
- n 产生Hermite-样条路径
- n 非常高效的计算

位置、速度和时间 (PVT 模式) [二次速度]

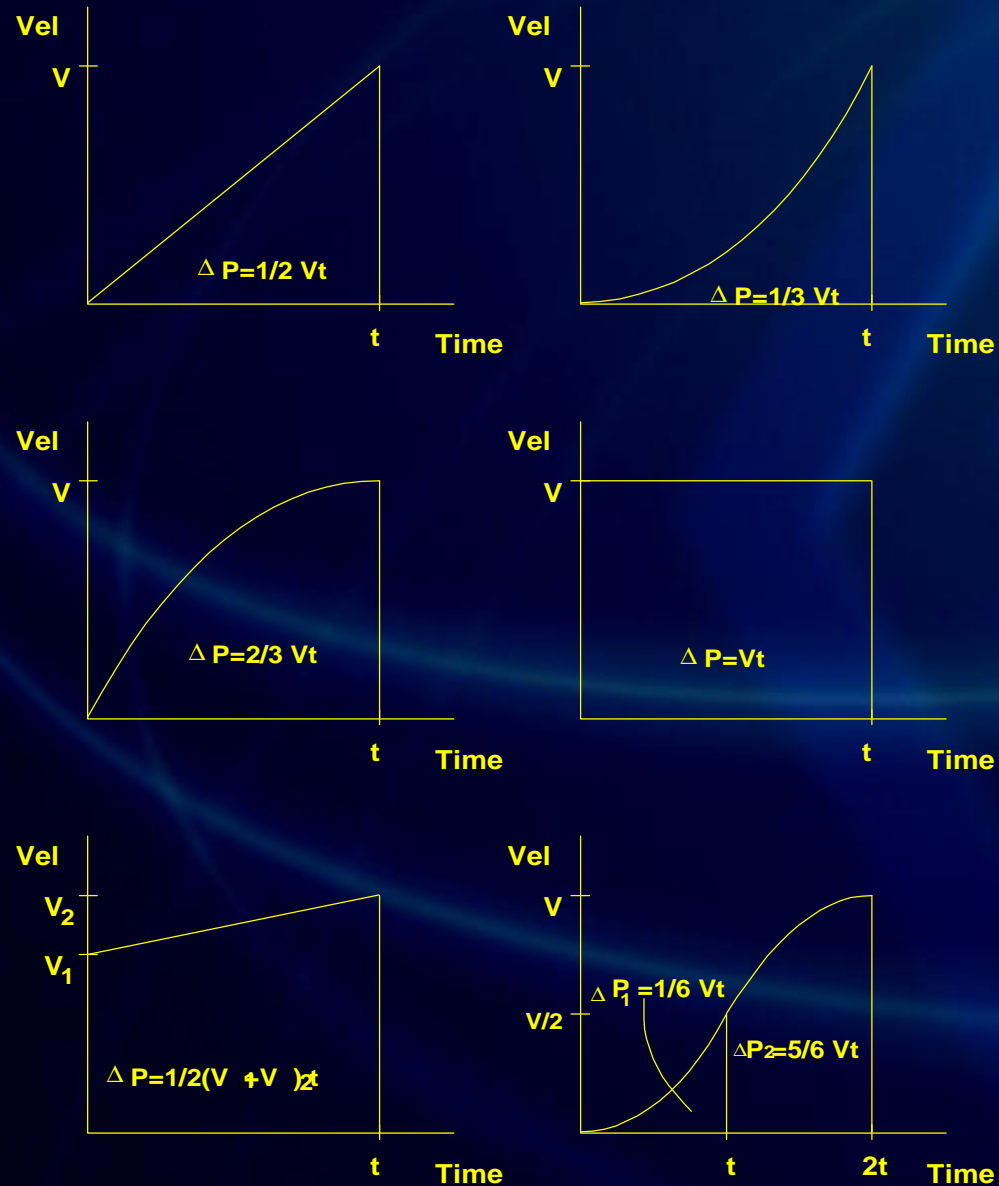


PVT模式：指定的位置和速度

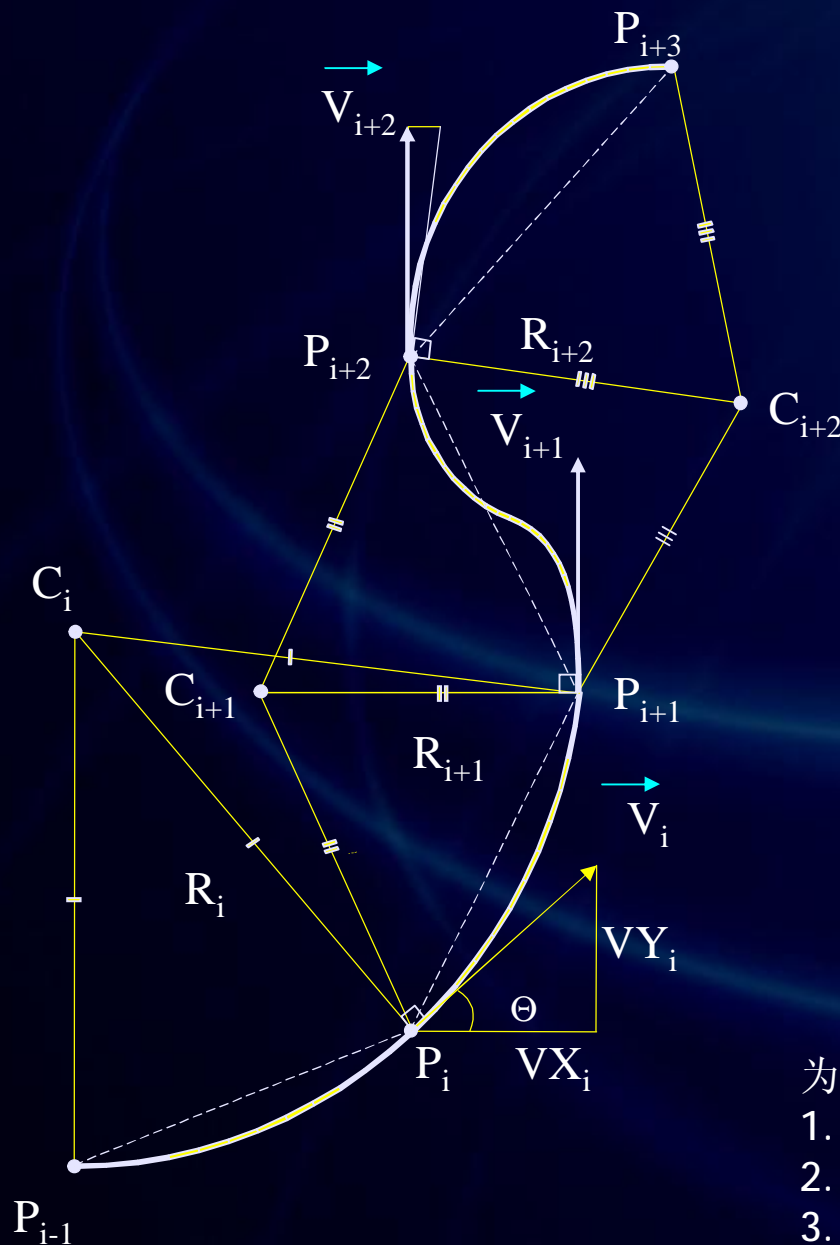


Move Program
INC
PVT200
① Y133.333:1000
PVT100
② Y100:1000
③ Y96.667:900
PVT200
④ Y140:500
PVT500
⑤ Y83.333:0

一般的PVT分段形状



PVT 模式曲线轮廓生成 (Hermite样条曲线)



为了计算P点的轴速度:

1. 找到 P_{i-1} , P_i 和 P_{i+1} 共同的中心点
2. 计算垂直于半径矢量的速度矢量
3. 分解速度矢量到各轴上

PMAC 子函数和子程序

子函数和子程序

- n GOSUB 300
 - n 跳转到：同一运动程序的N 300
- n CALL 500
 - n 跳转到： PROG 500, 在 (N0)的顶部
- n CALL 500.1
 - n 跳转到： PROG 500, label N10000
- n CALL 500.12
 - n 跳转到： PROG 500, label N12000
- n CALL 500.12345
 - n 跳转到： PROG 500, label N12345

- n 在线命令 B700
 - n 指向： PROG 700, (N0) 准备运行
- n 在线命令 B700.34
 - n 指向： PROG 700, N34000, 准备运行

传递参数给子函数

```
{PROG 1}  
CALL 500 D10E20
```

```
{PROG 500}  
READ (D,E)  
    sets Q104 to 10  
    sets Q105 to 20
```

A 参数读入到 Q101
B参数读入到 Q102
.
.
.
Y参数读入到 Q125
Z参数读入到 Q126

PMAC 子函数参数传递检查

- 如果 “nth” 个字母在最后的 READ语句中被传递，Q100的 (n-1)位设置为1
IF(Q100 & 16>0)
条件为真只有当“E” (5th 字母) 被传递 ($1=2^{5-1}$)
- 在READ开始，设Q100为0
- 成功读取参数后会在对应的Q100的位设置1

Z	Y		F	E	D	C	B	A	字母
0	0	...	0	1	1	0	0	0	Q100
25	24	...	5	4	3	2	1	0	位 #
2^{25}	2^{24}	...	32	16	8	4	2	1	位值 (十进制)
2^{25}	2^{24}	...	20	10	8	4	2	1	位置 (十六进制)

PROG1
CALL 500 D10 E20

PROG500
READ (D,E)

G, M, T, 和 D 代码

n 与RS-274 兼容的运动程序

CALL 1000.73000 **G 73** 相当于

CALL 1001.03000 **M 3** 相当于

CALL 1002.01000 **T 01** 相当于

CALL 1003.12000 **D 12** 相当于

G, M, T, 和 D 代码(续)

- n 泰道的软件提供缺省的标准G代码解释
- n 系统设计者在PROG 1000到PROG 1003内编写自定义例程
- n 程序员和机床操作员可以将一般的系统“改装”成 “G-代码” 机床
- n PROG 1000及以上的程序密码保护(同样PLC's 0 – 15也密码保护)

简单的G-代码程序

***** 部分程序文本 *****

;注释：在这部分程序中，不需要知道G-代码和M-代码如何执行

OPEN PROG 5 CLEAR	; 准备进入运动程序5
G17 G90	; XY平面，绝对运动模式
G97 S1800	; 设置主轴速度为1800rpm
F500	; 切削速度为500mm/min
G00 X10.00 Y5.00	; 快速运动到(10, 5)
M03	; 启动主轴
G04 P2.0	; 等待2秒
G01 Z0	; 降低切削速
X30.25 Y5.00	; XY线性运动
G03 X35.25 Y10.00 J5	; CCW圆弧运动
G01 X35.25 Y50.10	; 线性运动
G03 X30.25 Y55.10 I-5	; CCW圆弧运动
G01 X10.00 Y55.10	; 线性运动
G03 X5.00 Y50.10 J-5	; CCW圆弧运动
G01 X5.00 Y10.00	; 线性运动
G03 X10.00 Y5.00 I5	; CCW圆弧运动
G01 Z5 M05	; 切削上移，停止
G00 X0 Y0	; 返回到初始坐标
CLOSE	

简单的G-代码程序(续)

```
OPEN PROG 1000 CLEAR ; 准备进入缓存1000
RAPID RETURN ; G00快速方式(N0隐含)
N01000 LINEAR RETURN ; G01线性插补方法
N02000 CIRCLE1 RETURN ; G02顺时针圆弧模式
N03000 CIRCLE2 RETURN ; G03逆时针圆弧模式
N04000 READ(P) ; G04暂停P秒
IF (Q100 & 32768 > 0) ; P参数指定否?
DWELL (Q116*1000) ; PMAC指定以毫秒停顿
ENDIF
RETURN
N17000 NORMAL K-1 RET ; G17指定XY平面
N18000 NORMAL J-1 RET ; G18指定ZX平面
N19000 NORMAL I-1 RET ; G19指定YZ平面
N90000 ABS RET ; G90绝对方式
N91000 INC RET ; G91增量方式
N97000 READ(S) ; G97设置主轴速度
IF (Q100 & 262144 > 0) ; S(第19位)参数指定否?
I422=Q119/30 ; #4手动速度为cts/msec
ENDIF
RETURN
CLOSE
```

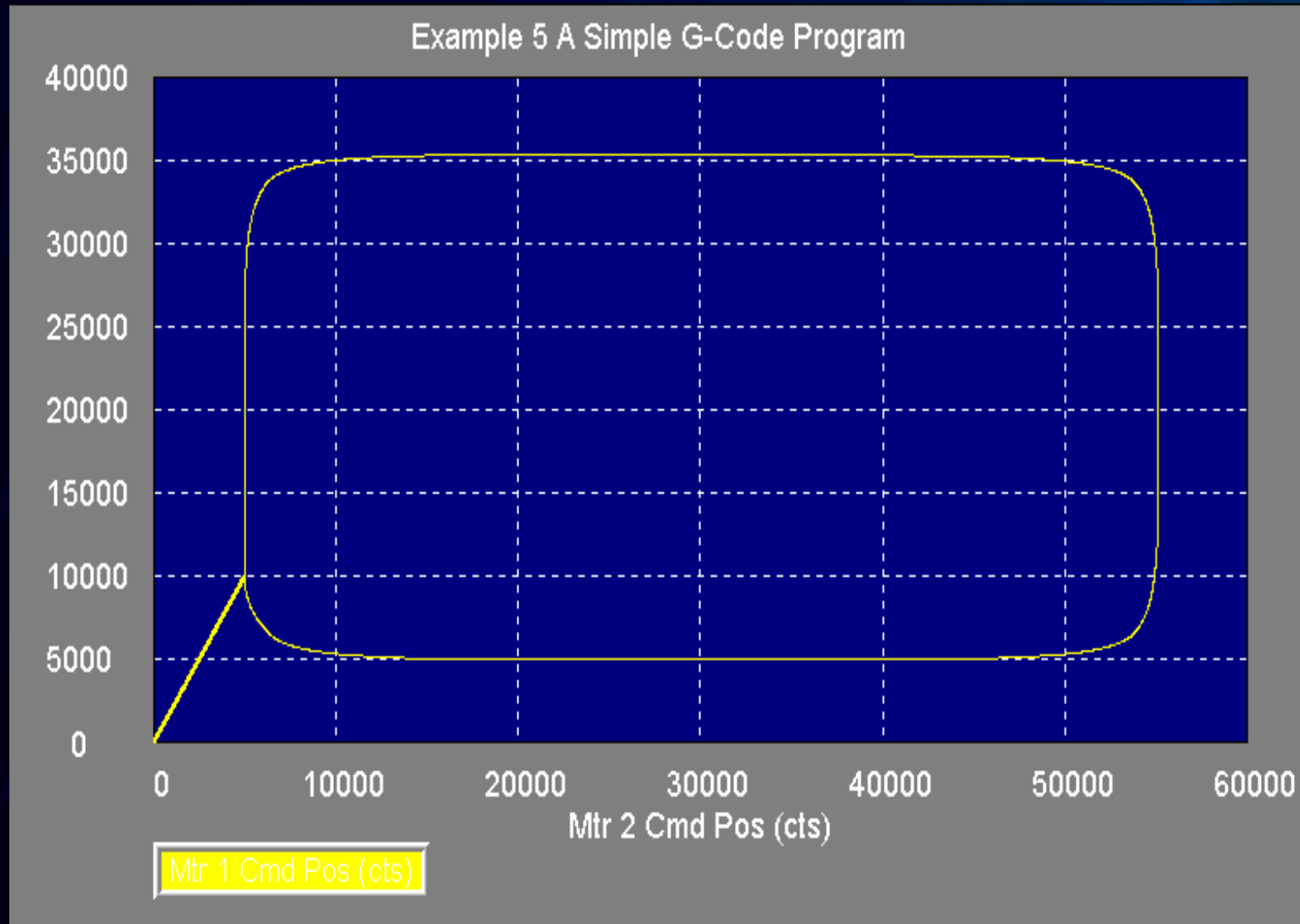
简单的G-代码程序(续)

```
OPEN PROG 1001 CLEAR ; 准备进入缓存1001
N03000 CMD "#4J+" RET ; 启动主轴顺时针(闭环)
N04000 CMD "#4J-" RET ; 启动主轴逆时针(闭环)
N05000 CMD "#4J/" RET ; 停止主轴
CLOSE
```

;运行此程序:

&1 B5 R ; 坐标系1, 指向程序5开始, 运行

简单的G-代码程序(续)



练习1： 电机运动控制应用

问题：

要求你使用PMAC进行运动控制应用，PMAC必须控制电机 #1，让它驱动一个丝杆运动10cm的距离，在这一位置暂停4 秒，然后返回到它的起始位置。每一步运动必须在2秒内完成。

硬件：

每转2000计数的编码器

5:1的齿轮减速

1cm/转 螺距丝杆

设lsx92=0，启用混合

练习2： 电机运动控制

问题：

为使应用通用，需要加入更多的要求，如练习1所示。

要求在下一张幻灯片中：

练习2： 电机运动控制 (续)

- n 期望的运动曲线已经变化。现在系统必须从它的零位置移动到10 cm，暂停4秒，然后在下一个2秒移动20cm，暂停1秒。然后必须在1.5秒内返回到0cm。
- n 现在系统成为一个多轴的项目。不是一个，而是两个同样的丝杆必须同时执行一定的曲线。
- n 为了尽量减小机床元器件的磨损，必须消除加速度的突变(jerk，跃度)。
- n 这个机器将被卖到欧洲的公司，在那里将会被写入一些新的程序。这些程序都是以厘米为单位，因此你的程序也必须用厘米为单位。
- n 系统也添加了进给率重载控制开关。当进给重载变化时，你的程序暂停时间也必须改变。这意味着在50%进给率重载时，1.5秒的暂停将成为3秒。

练习3：条件分支

问题：

欧洲的公司你的机器上已经具有许多成功的应用，现在老板有额外的研发经费，希望你在同一平台上编写其它的应用。2个输入开关已经连接到PMAC机器的输入1和输入2，希望利用这些开关控制运动舞台，如下所示：

- n 如果机器的输入1开通，舞台应该以1 cm/sec的速度从0cm移动到10cm，暂停1秒，然后以相同的速度返回到0cm，暂停100毫秒直到寻找下一个输入。

练习3： 条件分支(续)

- n 如果机器的输入2开通，舞台应该以2cm/sec的速度从0cm移动到-10cm，暂停1秒，然后以相同的速度返回到0cm，暂停100毫秒直到寻找下一个输入。
- n 如果机器的输入1和输入2都开通，或者都关断，舞台应该在0cm位置不移动，暂停100毫秒直到寻找下一个输入。

PMAC的PLC程序

PMAC的PLC程序

- n 像硬件PLC一样完成许多任务
- n 不考虑运动程序的状态，PLC进行重复、快速地周期计算

- n PLC可用于：

- n 监视输入
- n 设置输出
- n 改变增益
- n 监视卡的状态
- n 执行命令
- n 发送消息

```
CLOSE
DELETE GATHER
OPEN PLC 2 CLEAR
IF (M11=1)
    I130=10000
ELSE
    I130=8000
ENDIF
CLOSE
ENABLE PLC 2
```

} PLC语句

PMAC的PLC类型

n 前台 PLC (PLC0 or PLCC0)

- n 在实时中断运行 (RTI)
- n 重复率由 I8控制
- n 用于关键任务-尽可能的短!!

n 后台 PLC (PLC1-31 or PLCC1-31)

- n 在伺服周期之间运行
- n 重复速率由下列因素决定：
 - n 伺服频率
 - n 电机数和类型
 - n 运动程序的计算要求
 - n PLC 程序的长度和复杂程度

PLC & PLCC 区别


- n PLCC's是编译了的PLC's
- n 能更快的执行，因为如下原因：
 - n 消除解释时间
 - n 和PLC程序一样具有整数计算的能力
 - n 浮点运算在编译的PLC程序里比解释性的(一般) PLC程序要快2到3倍
 - n 整数运算 (包括布尔型) 在编译的形式下要快20到30 倍
 - n 一旦编译了，PLCCs 由编译它们的固件具体指定 (1.17版本以后)

后台PLC/PLCC 执行顺序

未编译的: PLC1, 2, 3...31

编译了的: PLCC1, 2, 3, 4...31

执行顺序:



PLC1
PLCC1, 2, 3, 4..
PLC2
PLCC1, 2, 3, 4..
PLC3
PLCC1, 2, 3, 4..

PLC0和PLCC0在更高的优先级，可以中断任何后台的PLC
相对解释的PLC，编译的PLC以更高的频率执行

PLC 编程控制

I5 = 0	不允许所有PLC's & PLCC's
I5 = 1	允许前台PLC's(PLC0 & PLCC0) 不允许后台PLC's
I5 = 2	不允许前台PLC's 允许后台PLC's (PLC's & PLCC's 1-31)
I5 = 3	允许所有的PLC's和PLCC's

- n 所有的PLCC程序在下载后都可单独地使能。
- n 所有存在的PLC's & PLCC's可在上电或复位时都使能。
- n 所有的由I5变量允许的PLC's & PLCC's，在上电或复位时都将立即运行或者I5变量允许后立即运行。
- n 所有存在的 PLC's可在上电或复位时保持使能，直到用户禁用。
- n PLC禁用后，只有你用I5允许它和使能它(Enable PLCxx)后才能运行。

PLC 编程控制 (续)

在线指令，运动程序和PLC程序表达式：

ENABLE PLC(C) n

DISABLE PLC(C) n

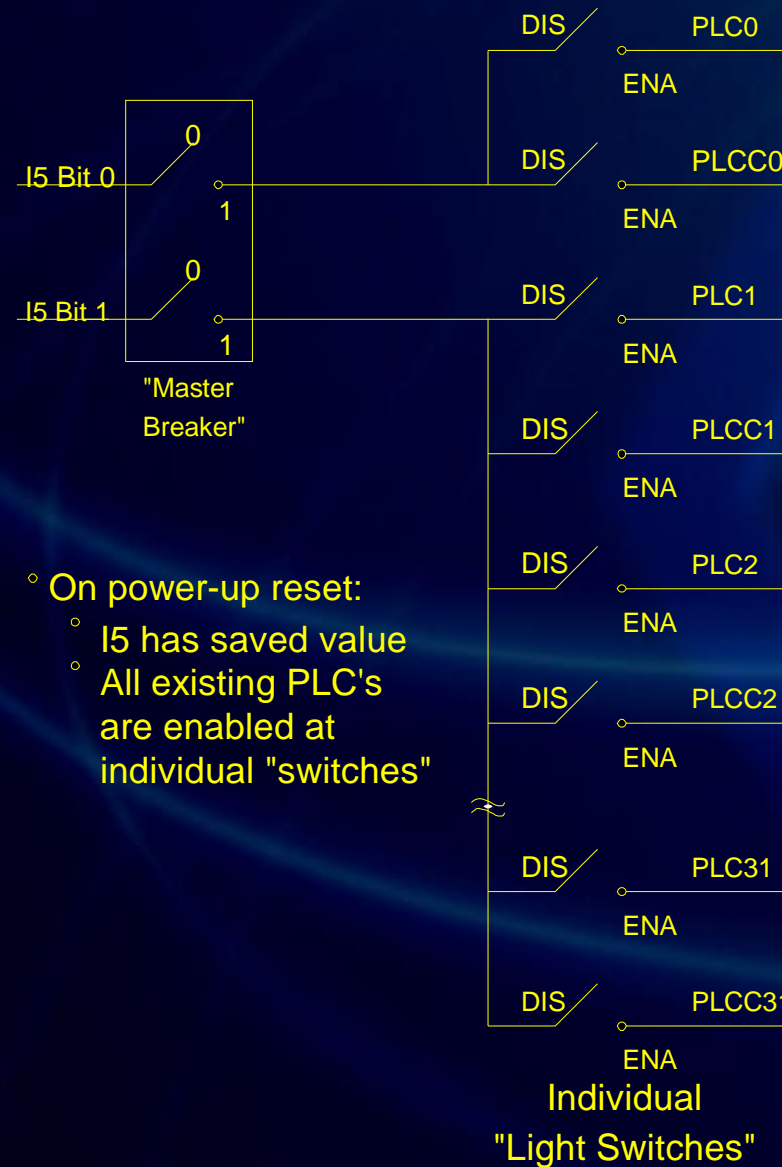
控制程序可以是独立的或成组的

<CONTROL-D> 禁用所有的PLC 程序

OPEN PLC n 禁用 PLC n

CLOSE 不启用 PLC n

PLC 编程控制



PMAC PLC的程序表达式

1. 条件表达式(可嵌套的)

IF({条件})

WHILE({条件})

AND({条件})

OR({条件})

where {条件}={表达式}{比较运算符 }{表达式}
[和/或{表达式}{比较运算符 }{表达式}...]

ELSE

ENDIF

ENDWHILE

2. 动作表达式

{变量} = {表达式}

COMMAND "{在线命令}"

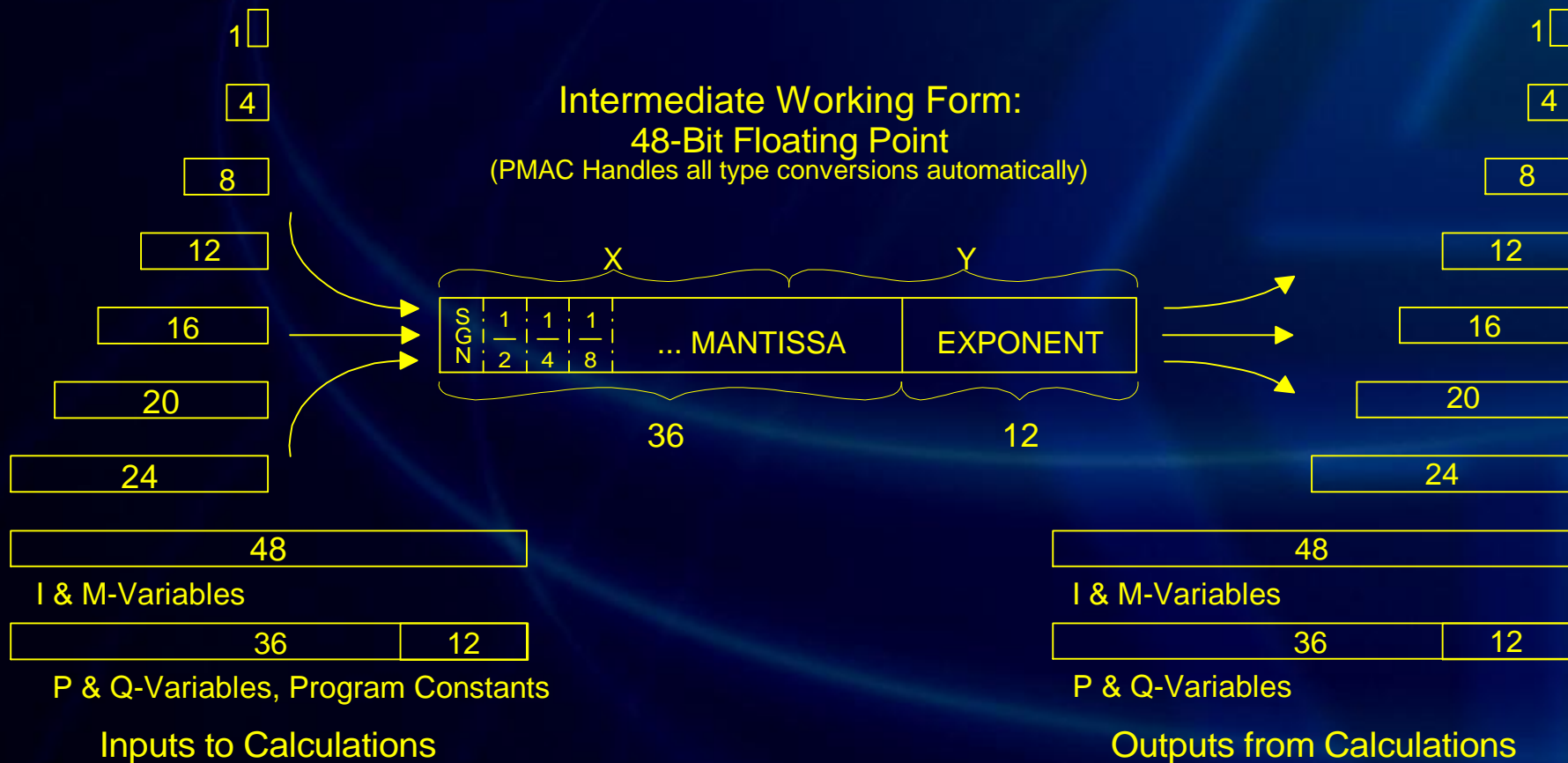
SEND "{消息}"

DISPLAY "{消息}"

ENABLE PLC {常数}

PMAC程序数学计算

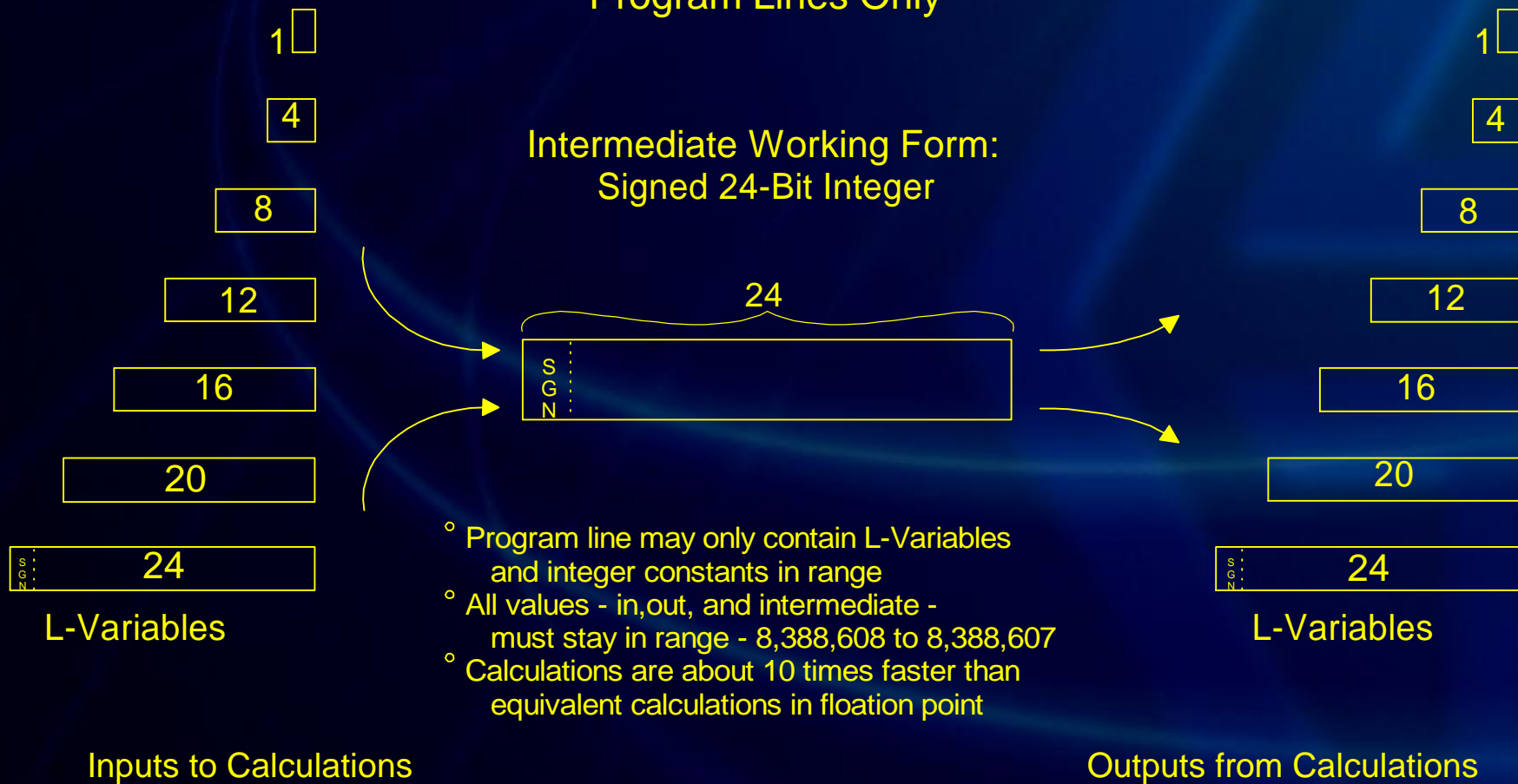
For All Motion and PLC Programs
Except Fixed-Point Compiled PLC Lines



PMAC程序数学计算(续)

Fixed Point Compiled PLC
Program Lines Only

Intermediate Working Form:
Signed 24-Bit Integer



编译PLC的L-变量

未编译的

```
M1->Y:$FFC2,8,1
M11->Y:$FFC2,0,1
M162->D:$002B
OPEN PLC1
CLEAR
IF (M11=0)
    M1=1
ELSE
    M1=0
ENDIF
P1=M162/I108/32
CLOSE
```

编译了的

```
L1->Y:$FFC2,8,1
L11->Y:$FFC2,0,1
M162->D:$002B
OPEN PLCC1
CLEAR
IF (L11=0)
    L1=1
ELSE
    L1=0
ENDIF
P1=M162/I108/32
CLOSE
```

PLC定数器延时

- n 由于DWEELL和DELAY命令只能用于运动程序中，因此PMAC定时器寄存器可用于PLC程序的时间延时。
- n 对于Turbo PMAC，每一个运行的坐标系有两个定时器变量运行：**Isx11**和**Isx12**.

C.S.#	I-变量	C.S.#	I-变量	C.S.#	I-变量	C.S.#	I-变量
1	I5111-I5112	5	I5511-I5512	9	I5911-I5912	13	I6311-I6312
2	I5121-I5212	6	I5621-I5612	10	I6021-I6012	14	I6421-I6412
3	I5311-I5312	7	I5711-I5712	11	I6111-I6112	15	I6511-I6512
4	I5411-I5412	8	I5811-I5812	12	I6211-I6212	16	I6611-I6612

PMAC例子: 若需要PLC程序中有1秒的延时

```
open plc 1 clear
```

```
.
```

```
I5111=(1000)*8388608/(I10)
```

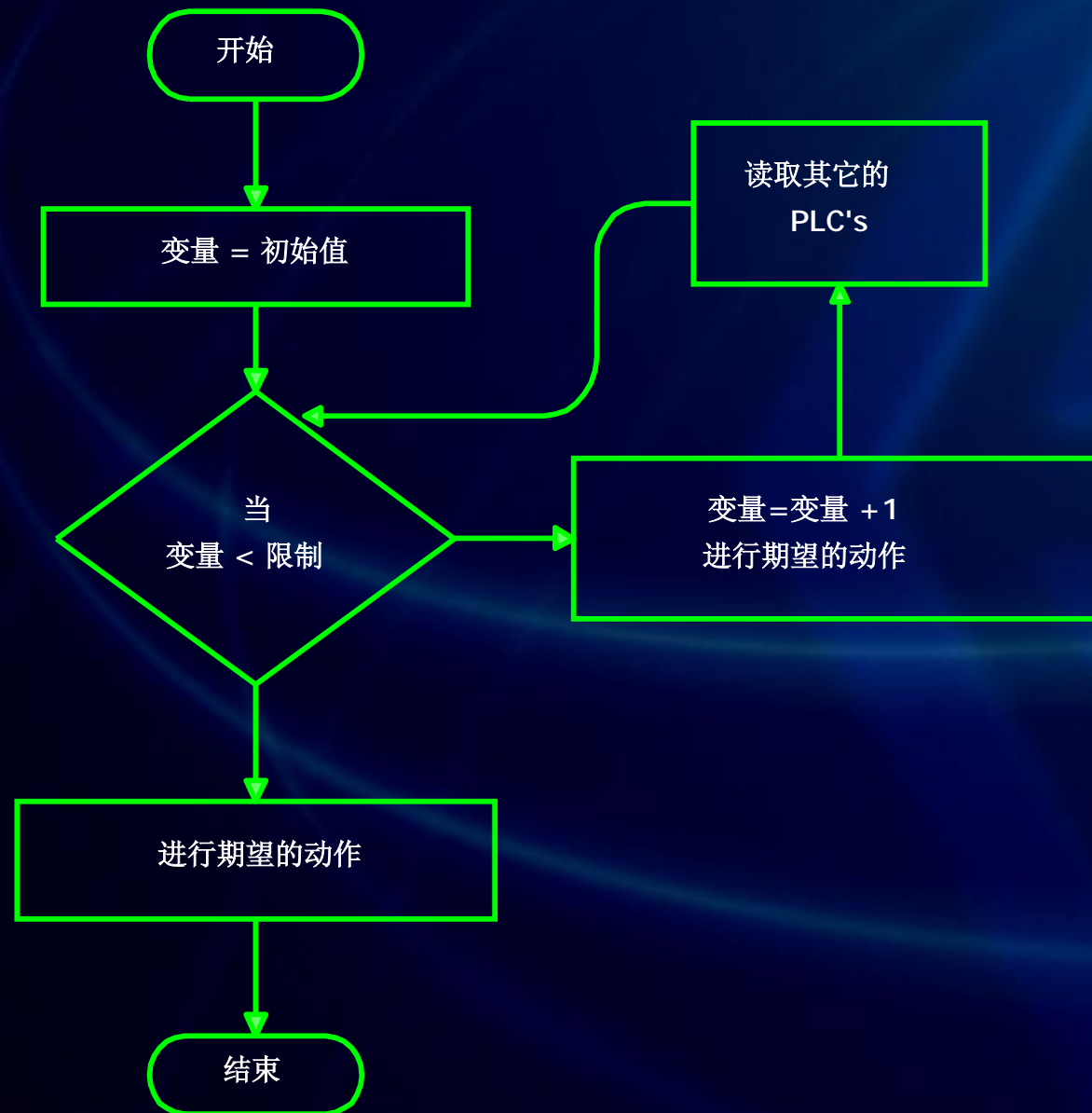
```
while (I5111>0)
```

```
endwhile
```

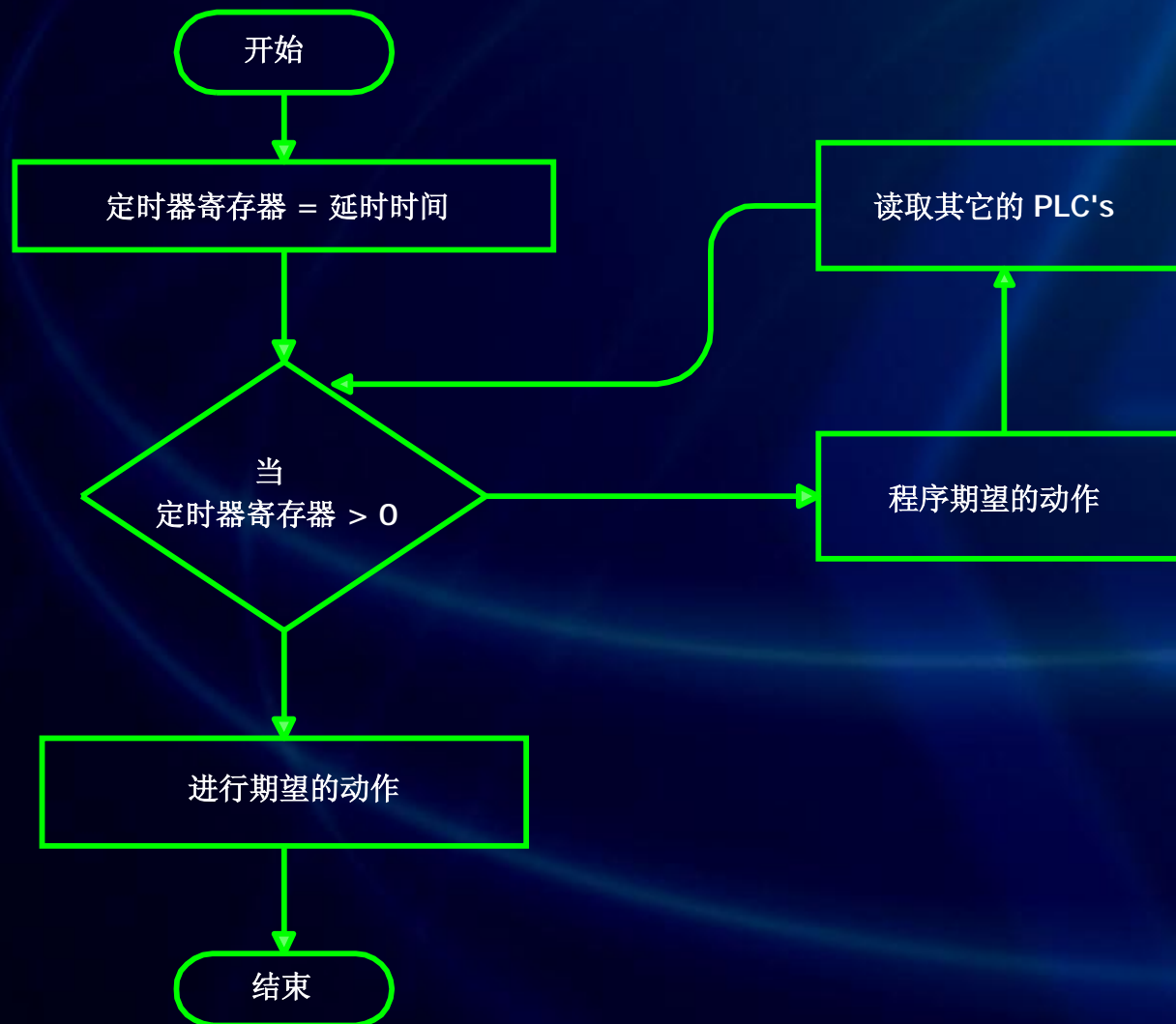
```
.
```

```
close
```


PLC定数器延时



PLC 定数器延时



PLC小测试

- n 说出PLC程序的3种类型.
- n 什么决定PLC程序是否运行?
- n 怎样使PLC's 2,3, 和10在上电或复位时自动运行?
- n 怎样知道某个命令是否可用于PLC程序?
- n 怎样使用PLC程序使Y-轴运动?
- n 怎样知道PLC是否启用?

PEWIN32PRO2 中的宏(MACROS)

宏允许用户以编程的目的，给PMAC变量或函数赋一个有意义的具体名称。

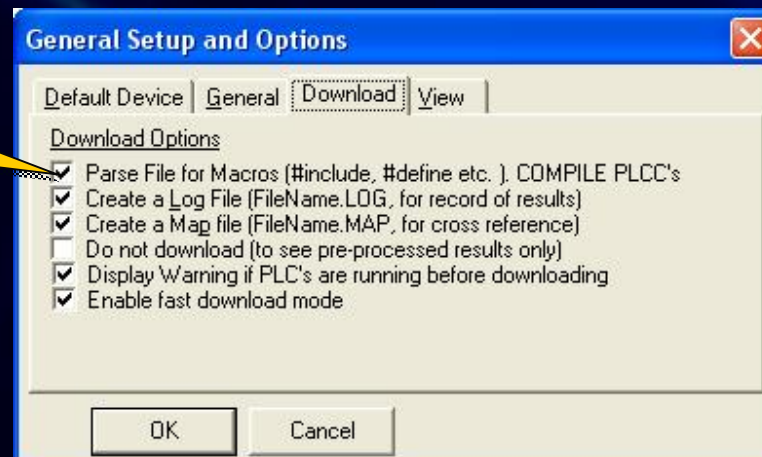
例如：

```
#define PUMP_PRESSURE P10
```

宏是PMAC Executive Program 软件而不是PMAC的特征。当文件中的Macros被下载到PMAC时，软件用 PMAC 命令或变量代替宏。

PMAC只能解释PMAC的命令和变量。

宏使用
选项



MACRO 定义

```
#define      E_STOP_IN_M      M900  
#define      LUBE_PUMP_M      M901  
#define      COOLANT_ON_M     M902
```

```
E_STOP_IN_M    ->Y:$FFC2,0  
LUBE_PUMP_M    ->Y:$FFC2,8  
COOLANT_ON_M   ->Y:$FFC2,9
```

在头文件中 (IO.h)

INCLUDE 语句

```
#include "{路径}"  
#include "C:\program files\delta tau\nc2.0\io.h"
```

作业 #1

- n a) 编写一个增加P-变量的PLC程序：利用while循环语句，使这个代码可以作为延时循环程序。
- n b) 编写一个可控制机器输出(例如从JOPTO接口)开通和关断的PLC程序。在关断和开通动作之间使用延时循环PLC程序。

作业 #2

- n 编写一个PLC程序：利用定时器，以选择的时间间隔开关灯。
- n 使用倒数计秒定时器完成这个程序。

作业 #3

- n 编写一个PLC程序：使得当按下开关时灯亮。
- n 使用闭锁，当按下瞬时开关时，灯亮；并一直保持到再按一下开关，灯灭。

作业 #4

- n 编写一个PLC程序：利用开关控制电机前向和反向手动。
- n 同时，当电机被赋予手动命令时，PLC程序应该发送SEND“ +/ -”语句，以指示手动的方向。

作业 #5

- n 编写一个PLC程序： 读取开关1演示软件的I/O仿真器，发送HOME命令。
- n 使用SEND “START” 语句，让PLC告知你： 什么时候回零开始？ 什么时候电机停止？

PMAC的内存 & I/O映射

PMAC 内存映射



Turbo PMAC的内存& I/O映射



注:

1. 56303 CPU / 56309 CPU
2. 标准/扩展的数据内存
3. 标准/扩展的BBRAM选项
4. 标准/扩展的DPRAM选项

PMAC的双端口RAM

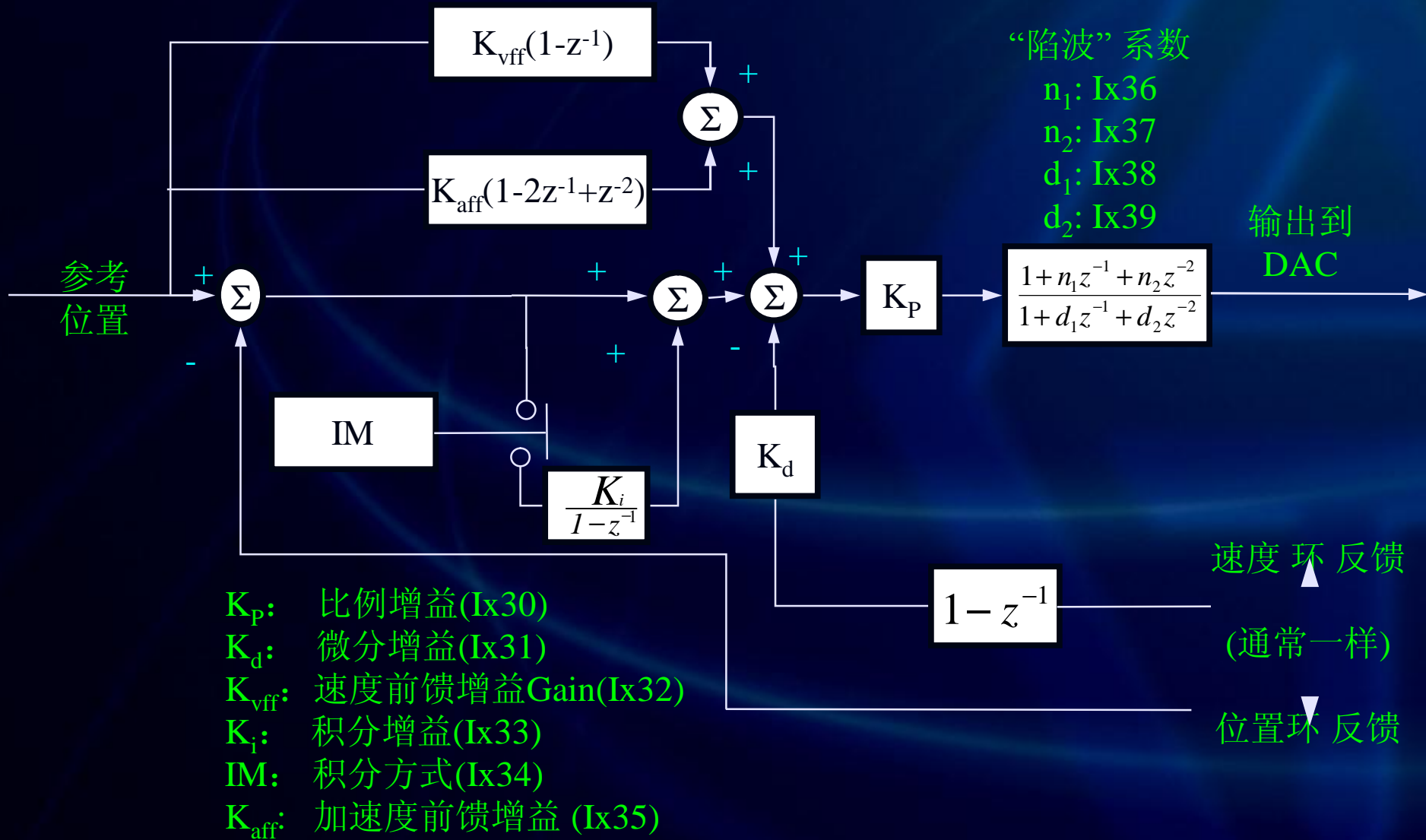
- n 双端口RAM、DPR或DPRAM, 用于在控制器，PMAC和主机(PC)需要同时访问和分享信息的场合。
- n 例： 控制系统需要PC处理获取的外部数据并被PMAC所分享.
- n 例： 重复的以非常快的速度向PMAC 下载位置数据 和/或 实时使用旋转 缓冲.
- n 例： 重复从PMAC获取非常快的状态信息.

PMAC的双端口RAM内存映射

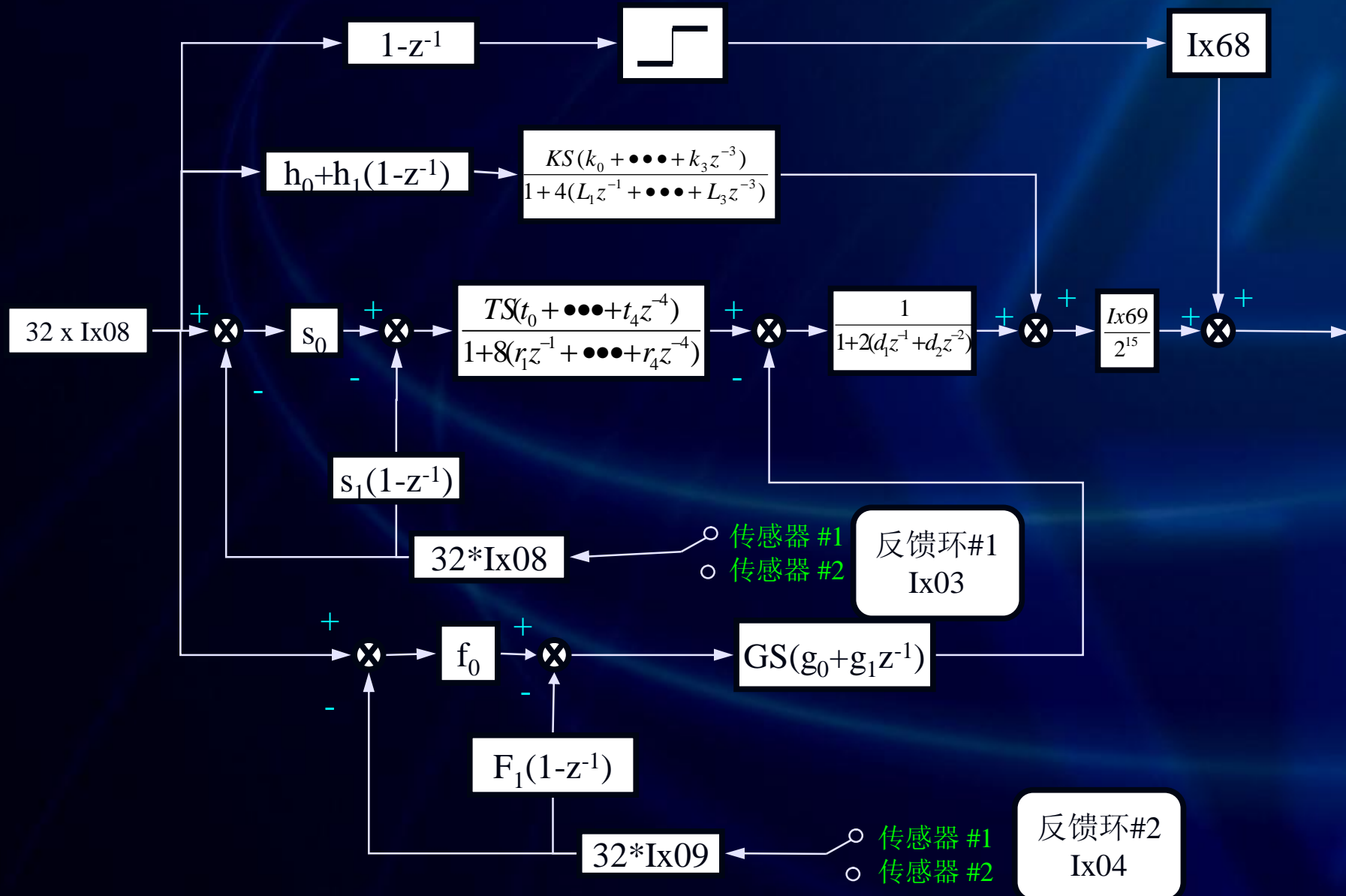
主机地址 偏移量		PMAC 地址	Turbo PMAC 地址
0x0000	控制 Panel Functions	\$D000	\$060000
0x0024	伺服 Data Reporting Buffer	\$D009	
0x0228	后台 Data Reporting Buffer	\$D08A	
0x062C	ASCII 命令 Buffer (Host to PMAC)	\$D18B	
0x06D0	ASCII响应缓存 (PMAC到主机)	\$D1B4	
0x07E8	指向变量-大小 缓存器	\$D1FA	
0x0810	Room for 变量-Size Buffers 1) 数据采集 2) 后台变量数据 3) Binary 旋转 Program	\$D240	
0x3FFC/ 0xFFFC	Open Use Space	\$DFFF	\$060FFF/ \$063FFF

PMAC的伺服控制算法

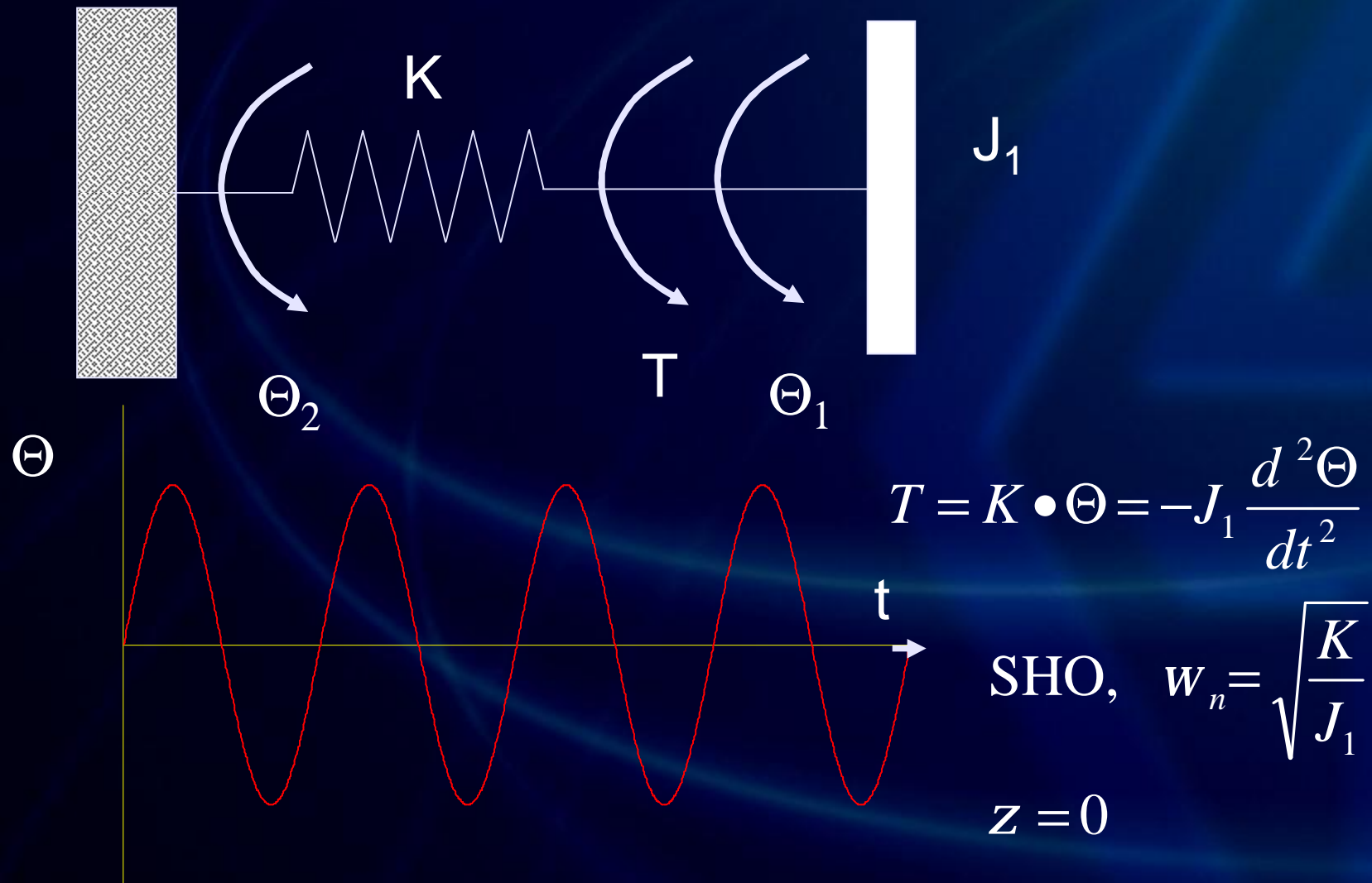
PMAC PID + 陷波伺服滤波器



PMAC 扩展的伺服算法

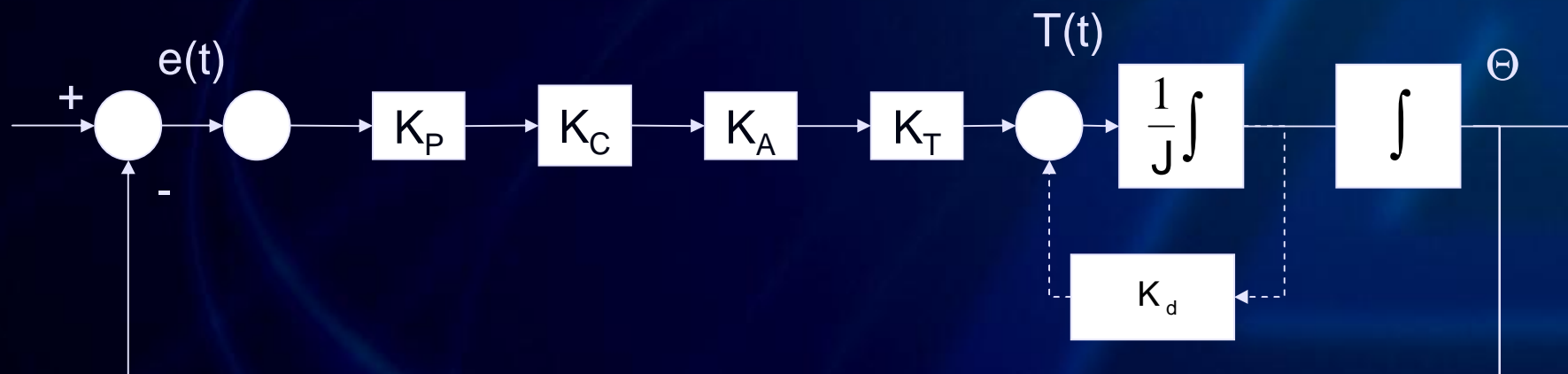


比例控制



SHO: 简谐振荡器

比例控制控制(续)



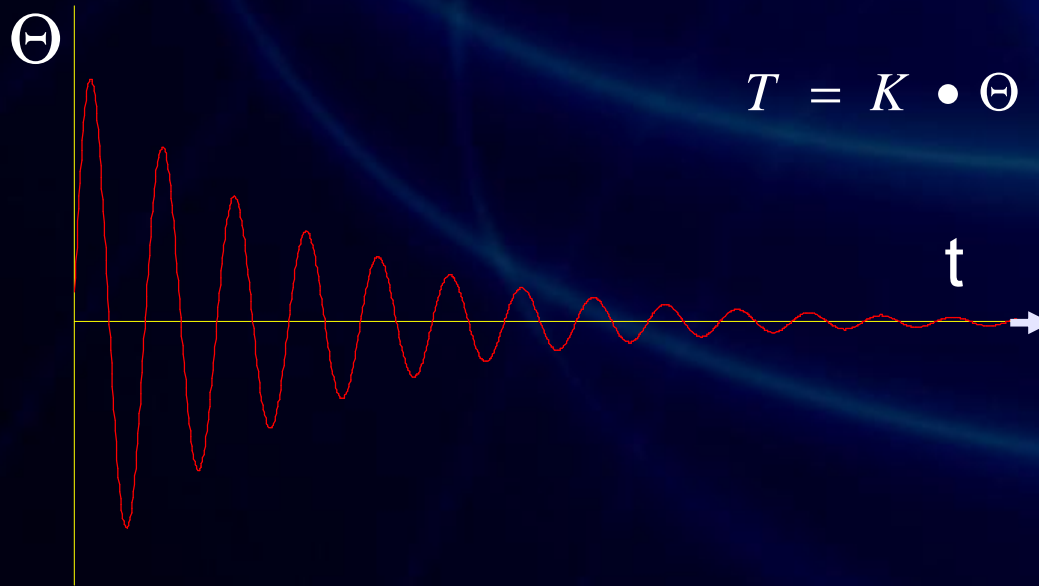
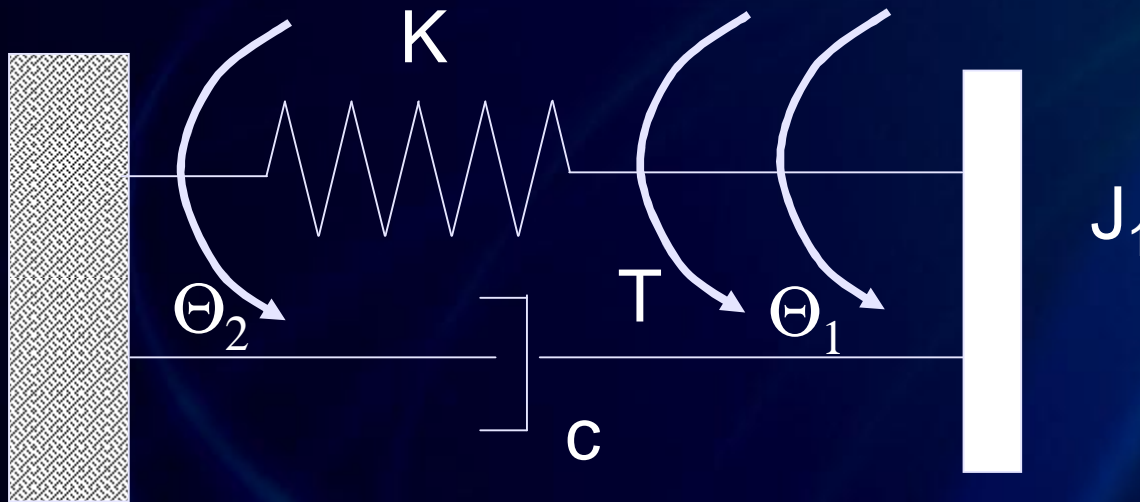
$$T(t) = K_p \cdot K_C \cdot K_A \cdot K_T \cdot e(t) = -K_p \cdot K_C \cdot K_A \cdot K_T \cdot \Theta(t) = J_1 \cdot \frac{d^2 \Theta}{dt^2}$$

这是个无阻尼系统

$$w_n = \sqrt{\frac{K_p \cdot K_C \cdot K_A \cdot K_T}{J_1}} \quad z = 0$$

- 因此比例增益 $K_p \rightarrow$ 弹簧刚度
- 高 $K_p \rightarrow$ 高刚度

微分控制

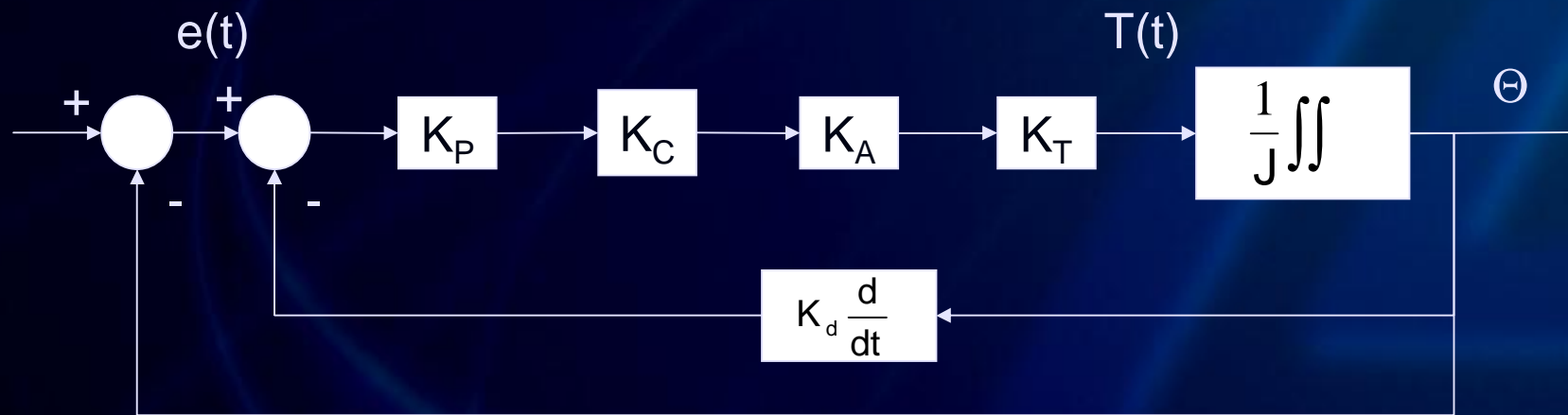


$$T = K \cdot \Theta + c \cdot \frac{d\Theta}{dt} = -J_1 \cdot \frac{d^2\Theta}{dt^2}$$

$$w_n = \sqrt{\frac{K}{J_1}}$$

$$z = \frac{c}{2} \sqrt{KJ_1}$$

微分控制(续)



$$T(t) = -K_p \cdot K_C \cdot K_A \cdot K_T \cdot \Theta(t) - K_p \cdot K_C \cdot K_A \cdot K_d \cdot \frac{d\Theta}{dt} = J_1 \cdot \frac{d^2\Theta}{dt^2}$$

这是个阻尼系统

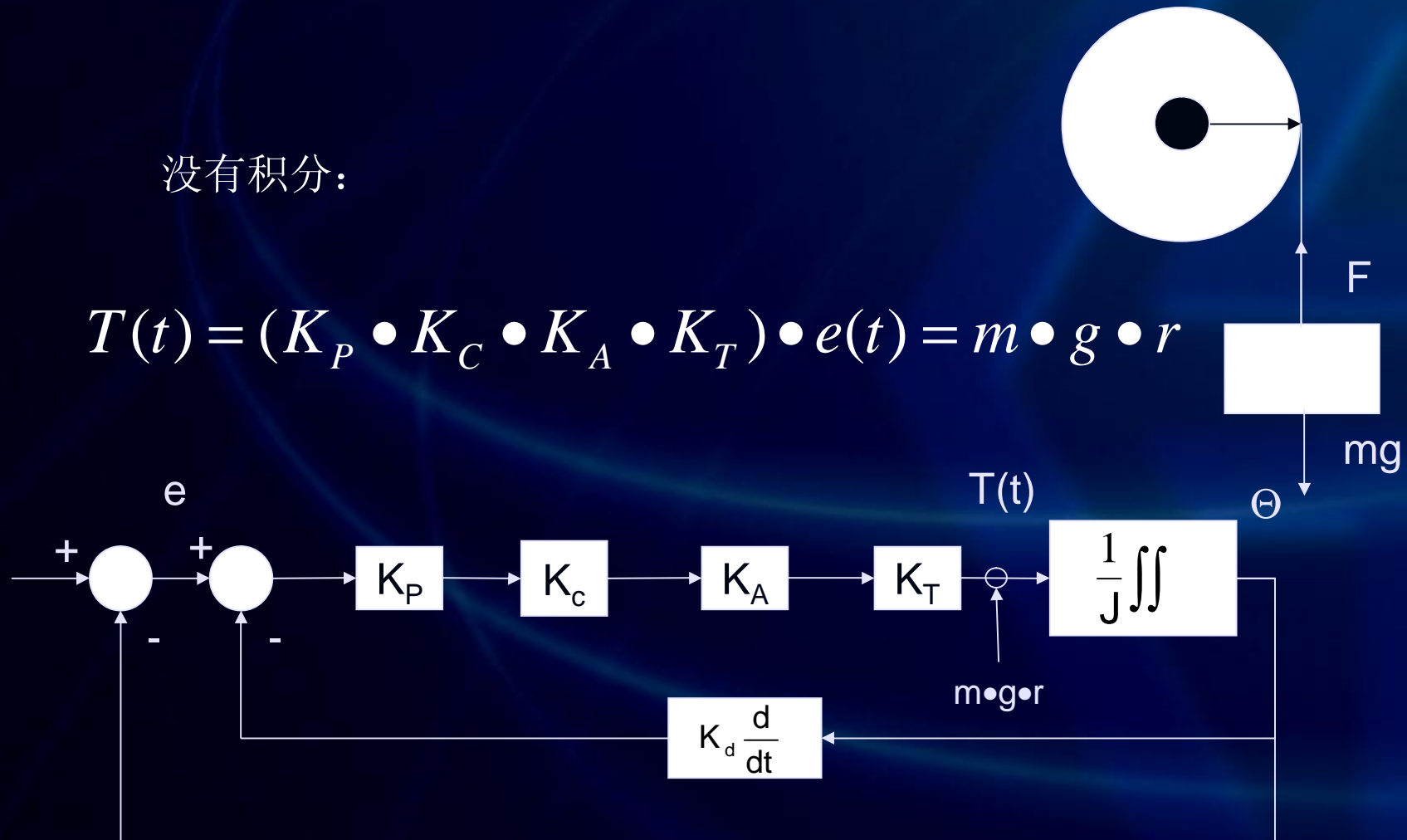
$$w_n = \sqrt{\frac{K_p \cdot K_C \cdot K_A \cdot K_T}{J_1}}$$

$$z = \frac{K_d}{2} \sqrt{\frac{K_p \cdot K_C \cdot K_A \cdot K_T}{J_1}}$$

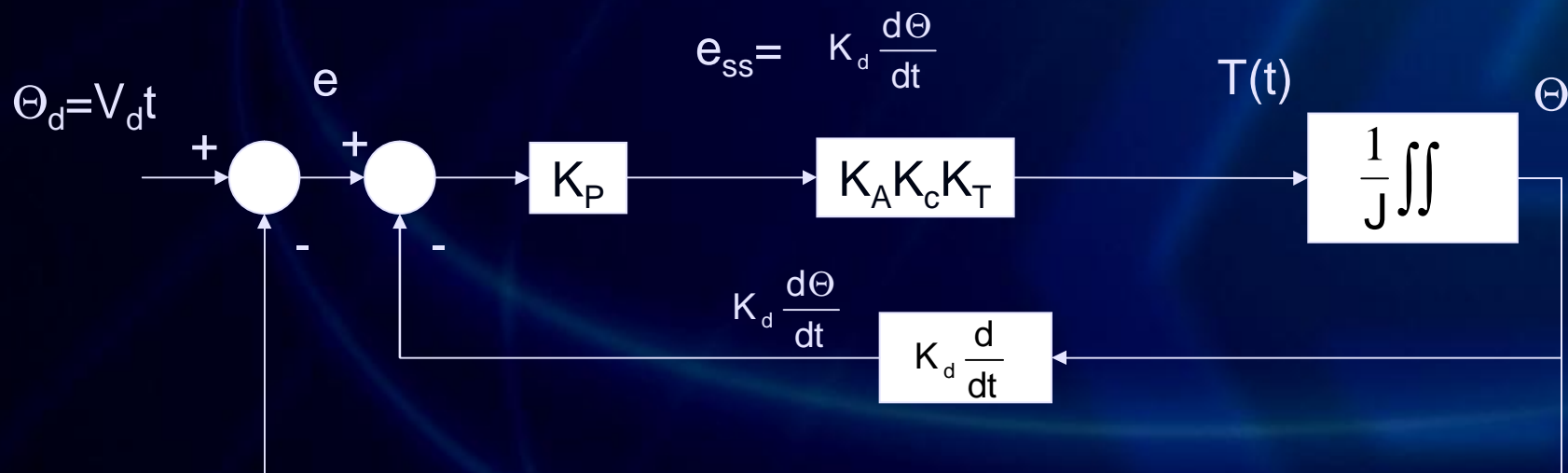
积分项

没有积分:

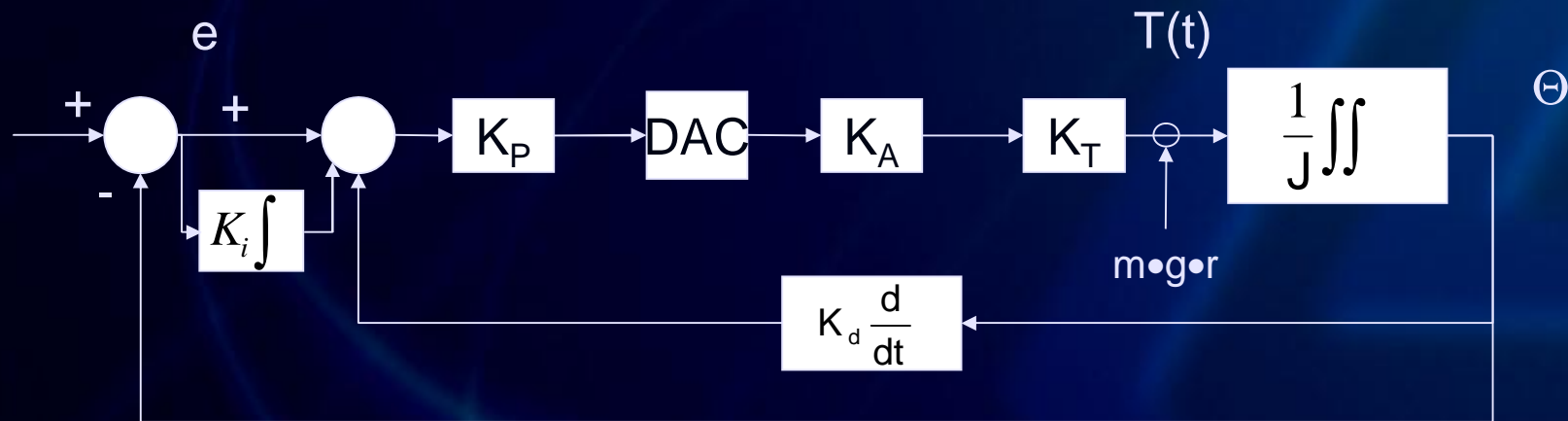
$$T(t) = (K_P \cdot K_C \cdot K_A \cdot K_T) \cdot e(t) = m \cdot g \cdot r$$



恒速轨迹跟踪的稳态误差



积分项(续)



$$T(t) = (K_P \cdot K_C \cdot K_A \cdot K_T) \cdot [K_i \int e(t) dt + d(t)] = m \cdot g \cdot r$$

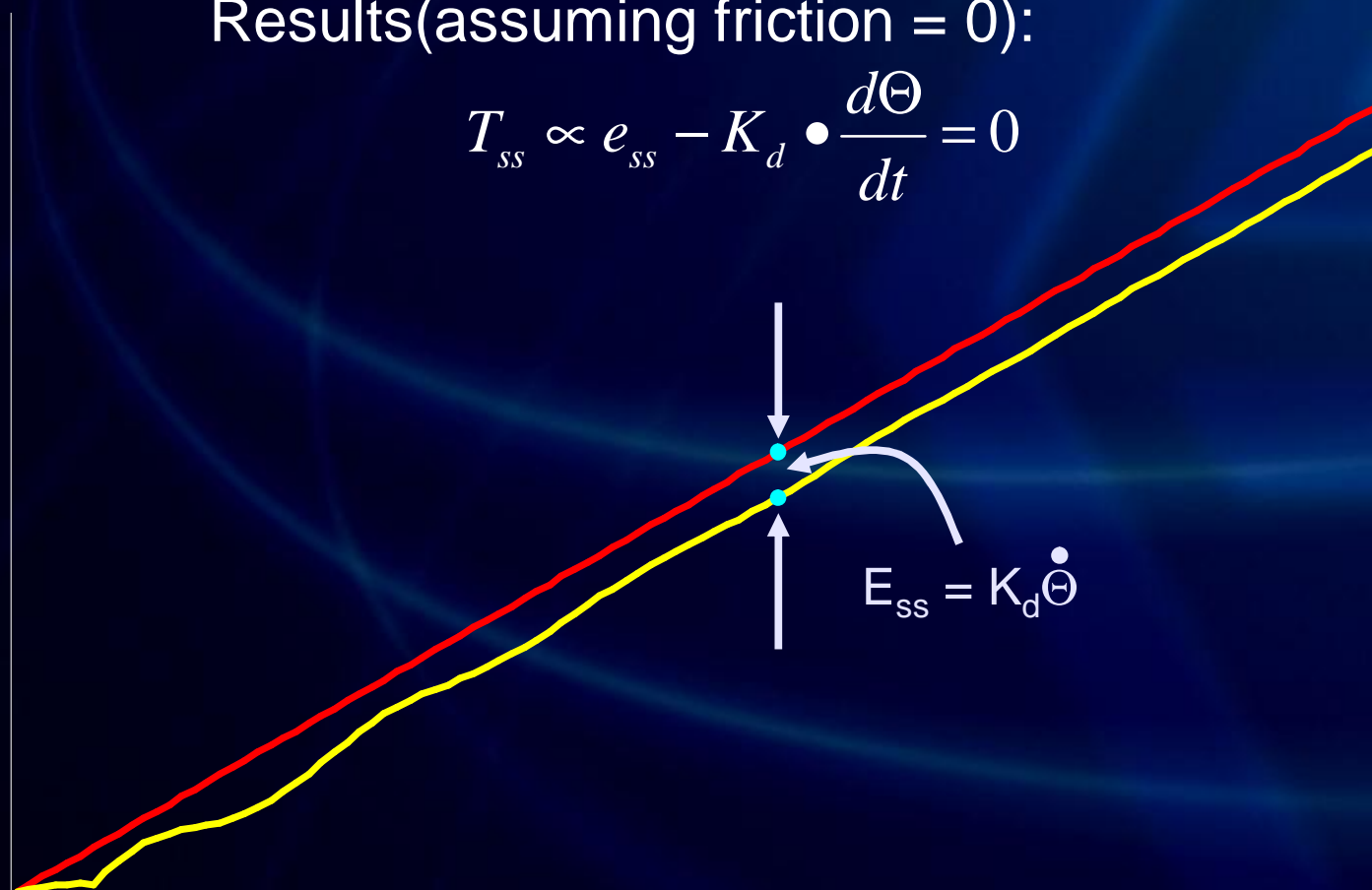
因此 $t \rightarrow \infty$, $e(t) \rightarrow 0$

恒速轨迹跟踪的稳态误差

斜坡输入: $\Theta_d = V_d \cdot t, \quad \frac{d\Theta}{dt} = V_d, \quad \frac{d^2\Theta}{dt^2} = 0$

Results(assuming friction = 0):

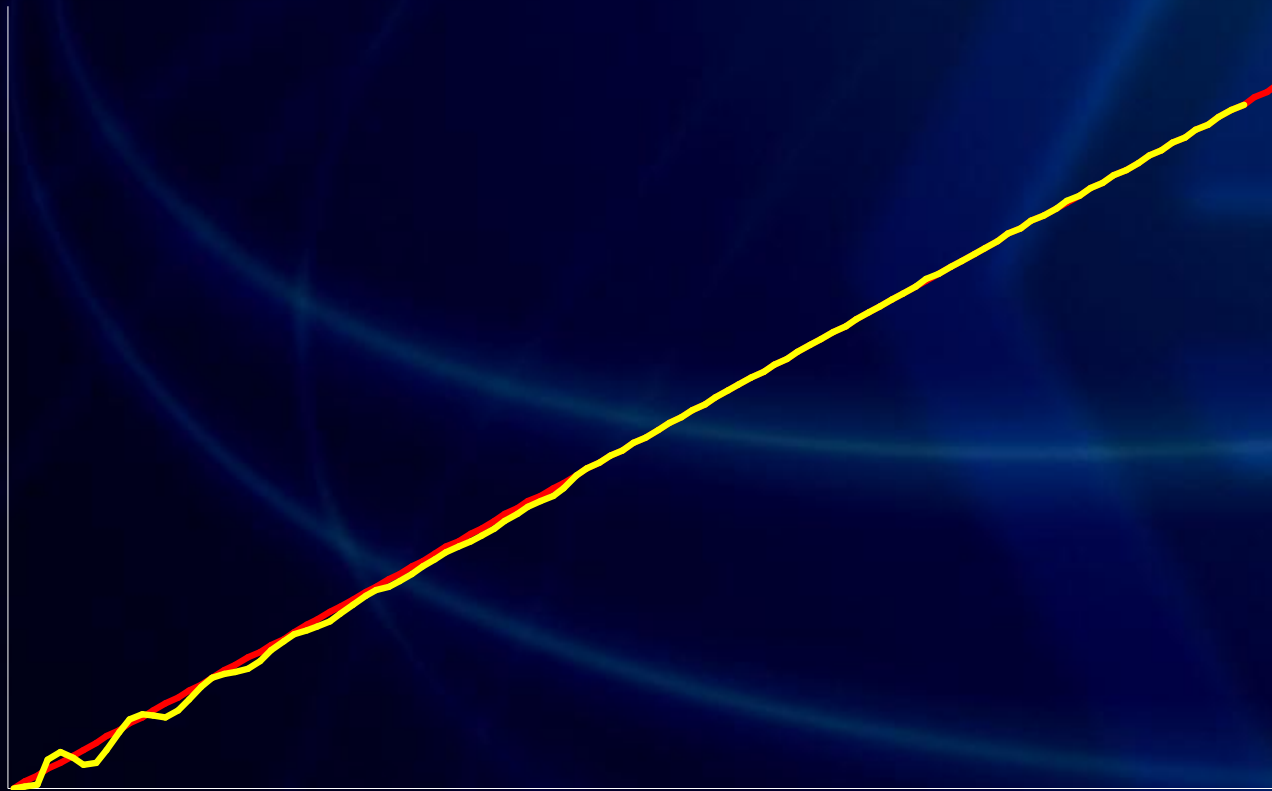
$$T_{ss} \propto e_{ss} - K_d \cdot \frac{d\Theta}{dt} = 0$$



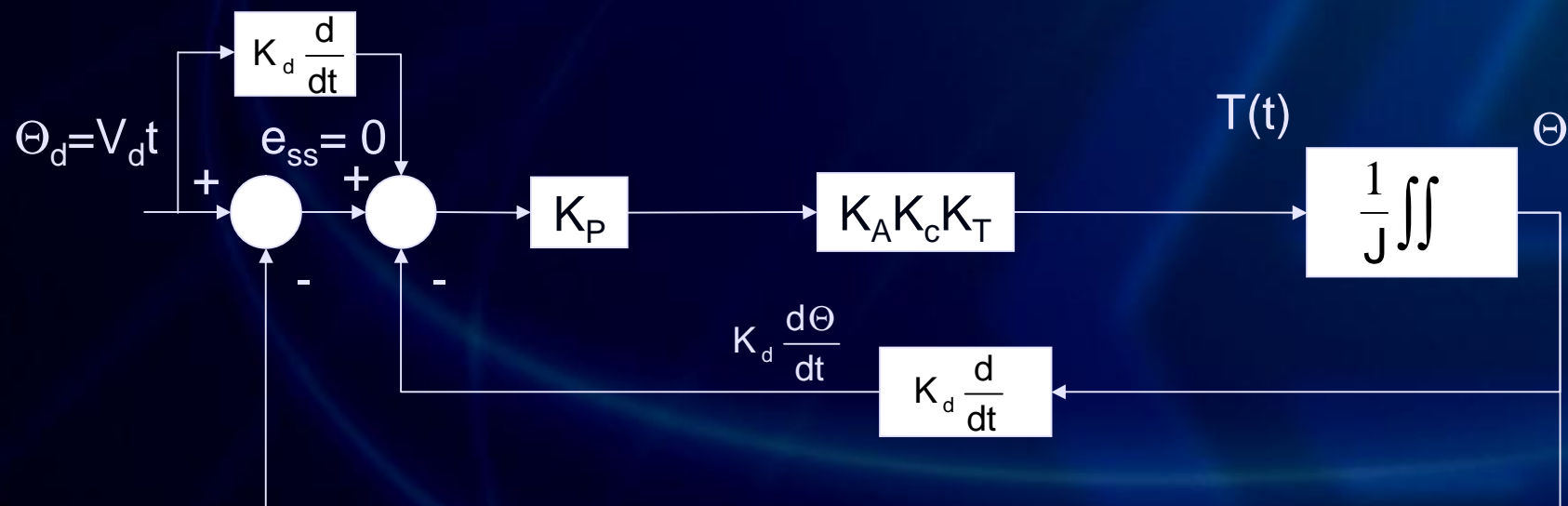
速度前馈的目的

选择 $K_{vff} = K_d \rightarrow e_{ss}=0$

由于恒速跟踪，无稳态误差



速度前馈



加速度前馈的目的

通常，轨迹含有高阶时间函数：

例如：恒定突变轨迹

$$\Theta_d(t) = c_0 + c_1 \cdot t + c_2 \cdot t^2$$

通过选择，

$$K_{aff} = \frac{J_1}{K_P \cdot K_C \cdot K_A \cdot K_T} \rightarrow e(t) = 0$$

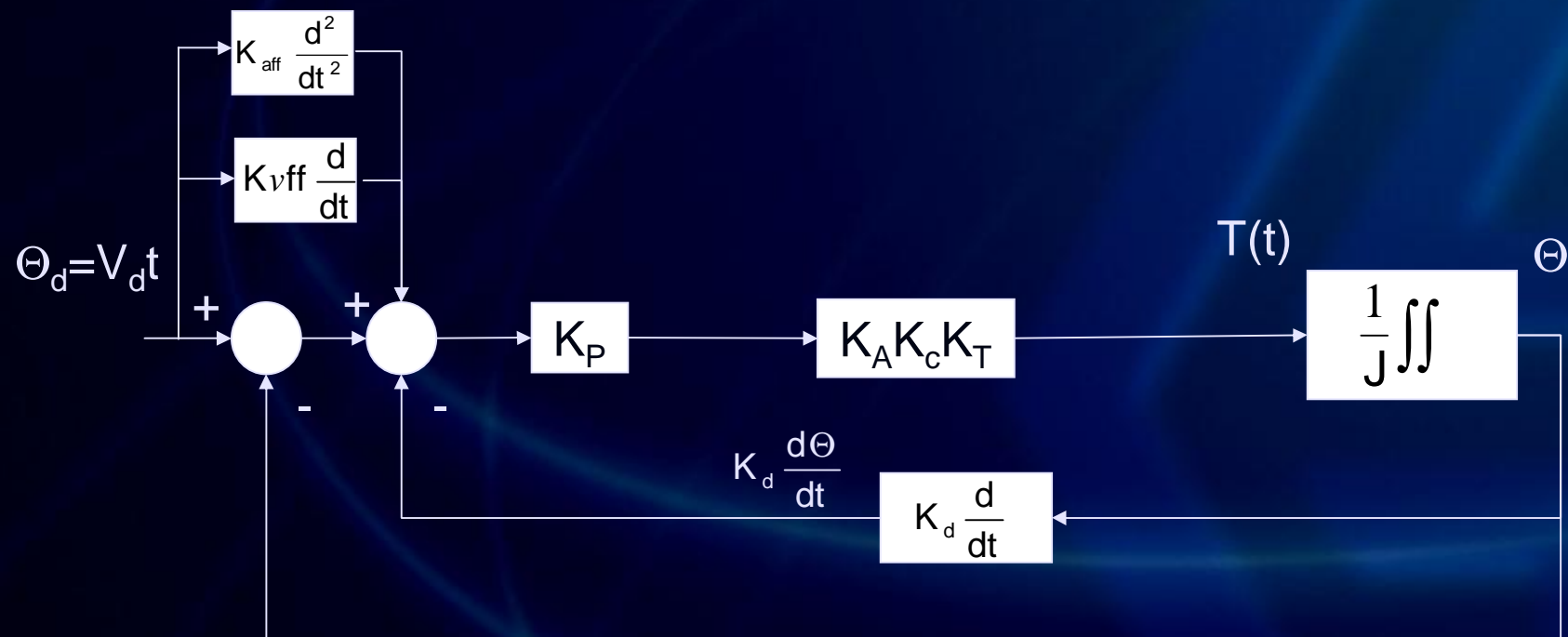
对于理想的系统，这一结果无跟踪误差。

由于，

$$T(t) = \left(\frac{J_1}{K_P \cdot K_C \cdot K_A \cdot K_T} \right) \frac{d^2 \Theta_d}{dt^2} (K_P \cdot K_C \cdot K_A \cdot K_T) = \frac{J_1 \cdot d^2 \Theta_d}{dt^2}$$

$$\frac{d\Theta_d}{dt^2} = \frac{d^2 \Theta}{dt^2} \rightarrow \Theta_d = \Theta \rightarrow \text{无跟踪误差}$$

加速度前馈



伺服环 调整因子 变量

- n lx59: 用户编写的伺服/相位使能
 - 0: 使用标准PID相位算法
 - 1: 使用自定义伺服, 标准相位算法
 - 2: 使用标准PID, 自定义相位算法
 - 3: 使用自定义伺服, 相位算法

- n lx60: 伺服环周期扩展
 - 每个(lx60+1)伺服中断闭环
 - 用于慢速、低分辨率的轴
 - 用于过程控制“轴”

伺服环 调整因子 变量 (续)

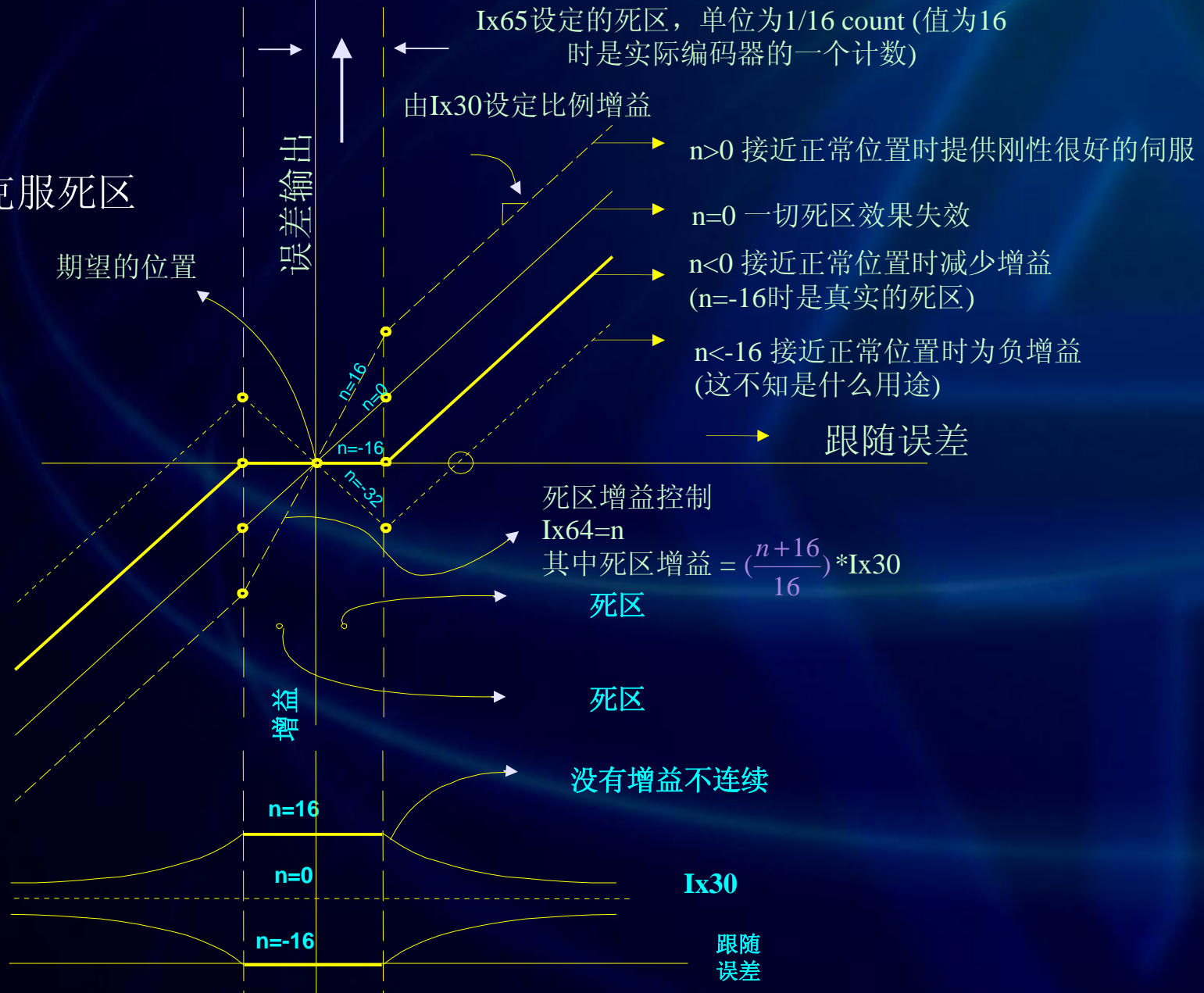
- n** lx63: 积分限制
累计最大积分误差
若为负，积分器饱和时停止电机

- n** lx64: “死区增益”
修正增益的范围

- n** lx65: 死区范围
修正增益的范围

PMAC的死区补偿

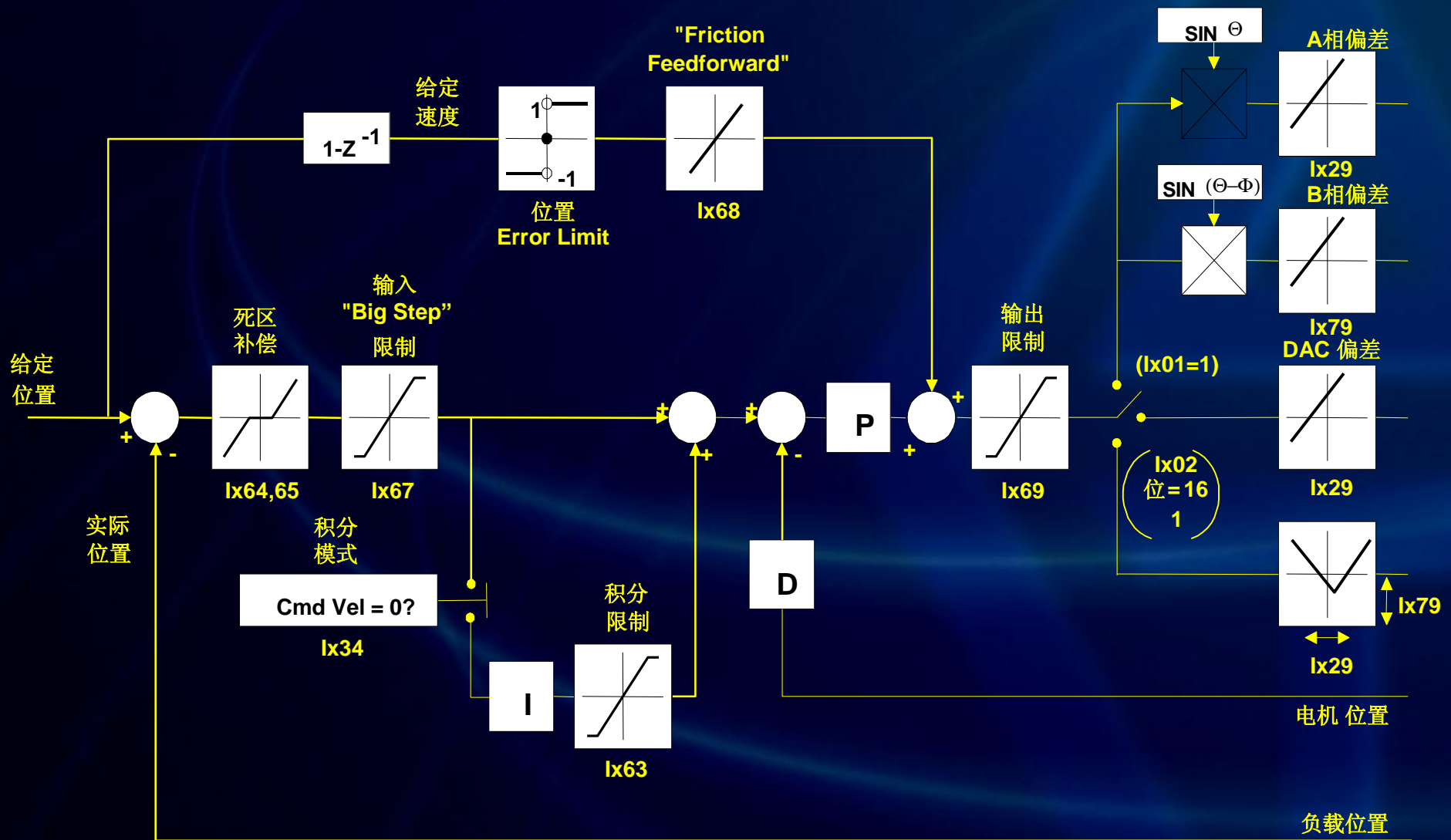
创建或克服死区



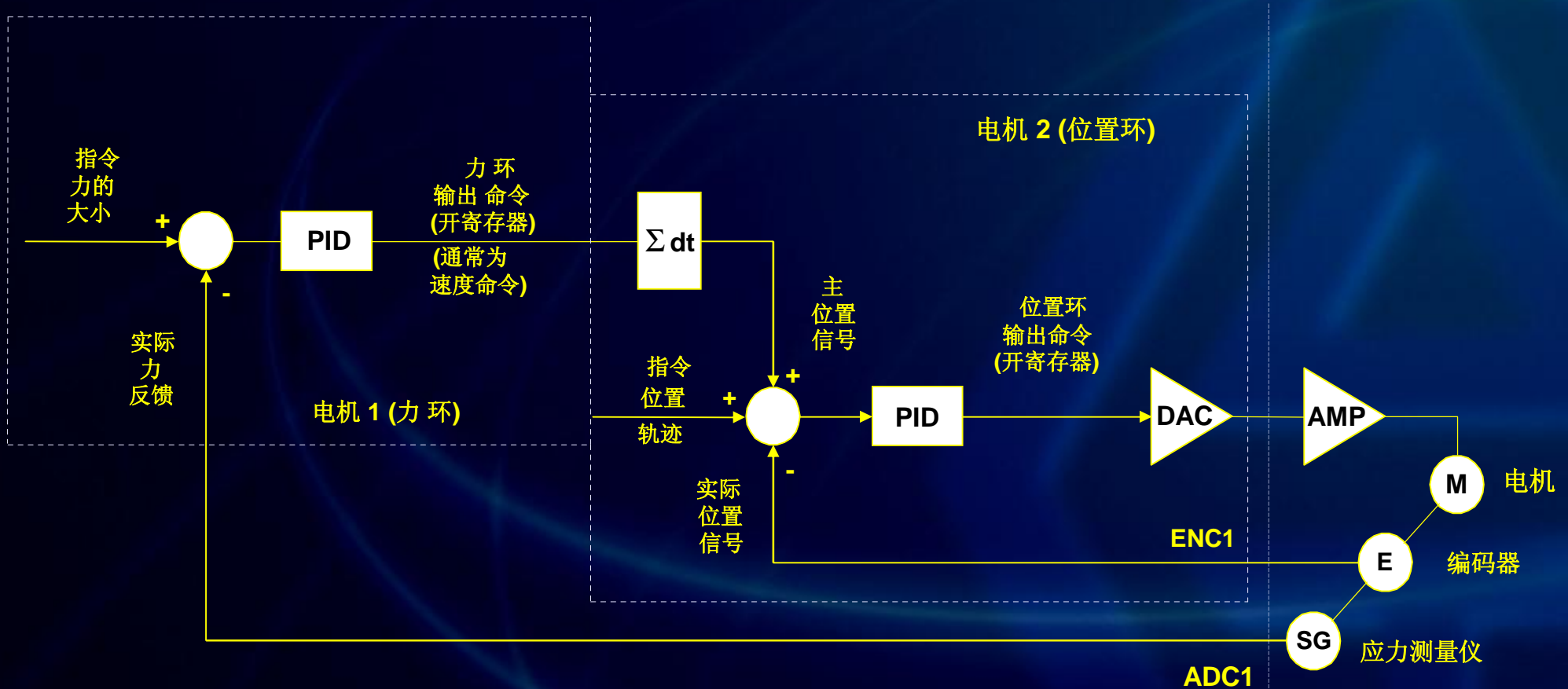
伺服环调整因子变量 (续)

- n lx67: 位置误差限制
限制滤波器所接受的最大轨迹误差
- n lx68: 摩擦前馈
补偿干 (Coulumb) 摩擦
- n lx69: 输出命令(DAC)限制
限制滤波器输出
换相时为转矩 (电流)限制

PMAC PID 伺服环 调整因子



在位置环外进行外力闭环



I102 = \$07F0 (Open Y 寄存器 \$07F0)
 I103, I104 = \$072A (ADC1转换)
 I105 = \$072B (ADC2 转换)

I202 = \$C003 (DAC1)
 I203, I204 = \$0720 (ENC1)
 I205 = \$1072D 用于对力积分
 命令 叠加到
 位置轨迹

转换表做积分

示例
 设置: WY\$072C, \$5007F0,\$0
 结果 X:\$072D

转换表必须处理所有
 使用的编码器和ADC

PMAC的时基控制

什么是PMAC的时基控制？

- n 在每一个伺服周期利用插补函数计算命令位置：
$$CP_n = CP_{n-1} + CV_n \Delta t$$
- n Δt 表示每伺服周期流逝的时间
24-位寄存器： $2^{23} = 1$ 毫秒
- n 伺服周期之间的物理时间由硬件设置
- n 改变 Δt 寄存器的值使之异于流逝的真实事件提供“时基 控制”
 (“进给 重载”)

PMAC 的时基控制变量

- n I10 伺服中断时间
 - n 告诉插值器伺服周期的真实时间
 - n 2^{23} (8M) 个计数相当于 1毫秒
 - n 默认的I10 = 3,713,707 (442 μ sec)
- n Isx93 坐标系时基地址
 - n 指出用哪个寄存器来读取时基值
 - n 寄存器是以I10为单位

有哪些普通时基源？

- n % 命令 (缺省)
- n 操作者调节了 A/D 或 V/F的输入
- n 操作者通过 PLC程序切换了开关
- n 主编码器计数频率
- n 外部时钟频率
- n 处理变伺服环输出

时基主命令

- n % {常数} 命令为寻址的坐标系把 $(10 * \{常数\} / 100)$ 的值放入寄存器。
- n %100为“实时”
- n % query 命令显示当前时基值 (与源无关)
- n lsx93 = 默认值, 这一值用于时基 Δt
- n 每lsx94 个伺服周期, 时基值随着新的命令改变

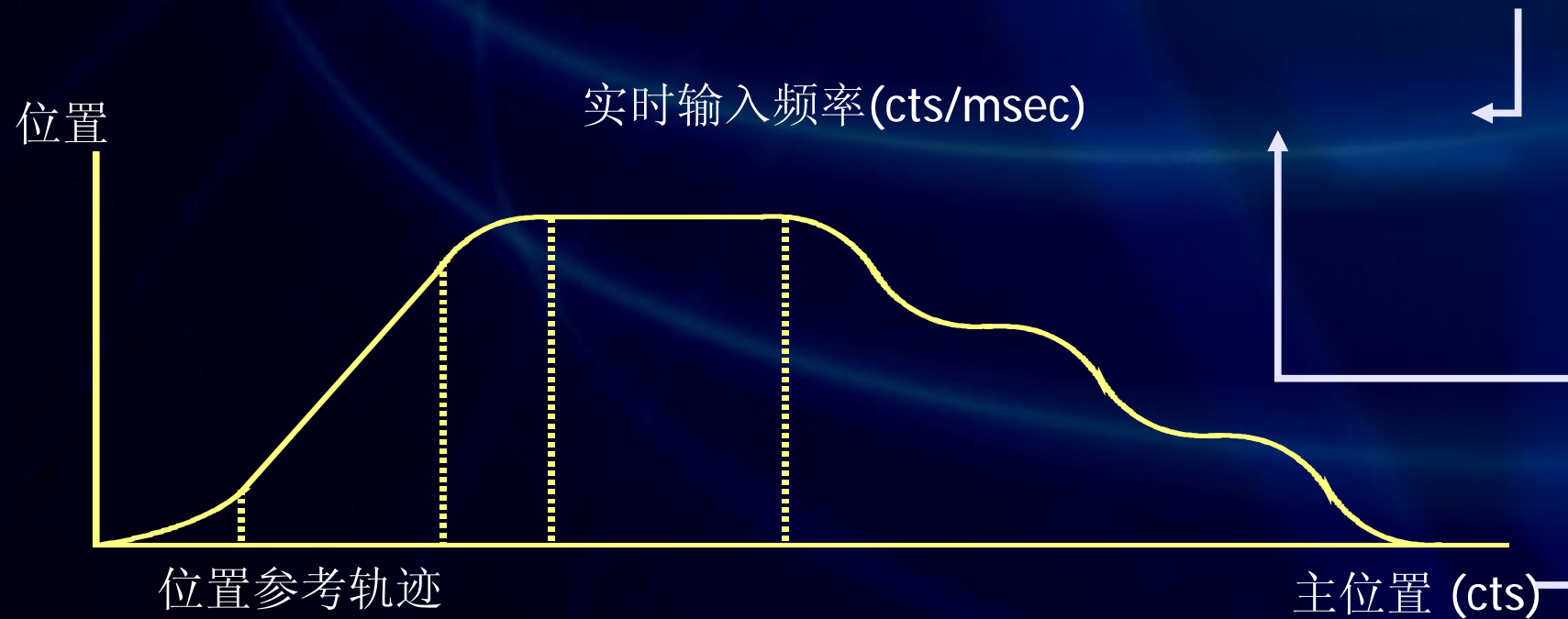
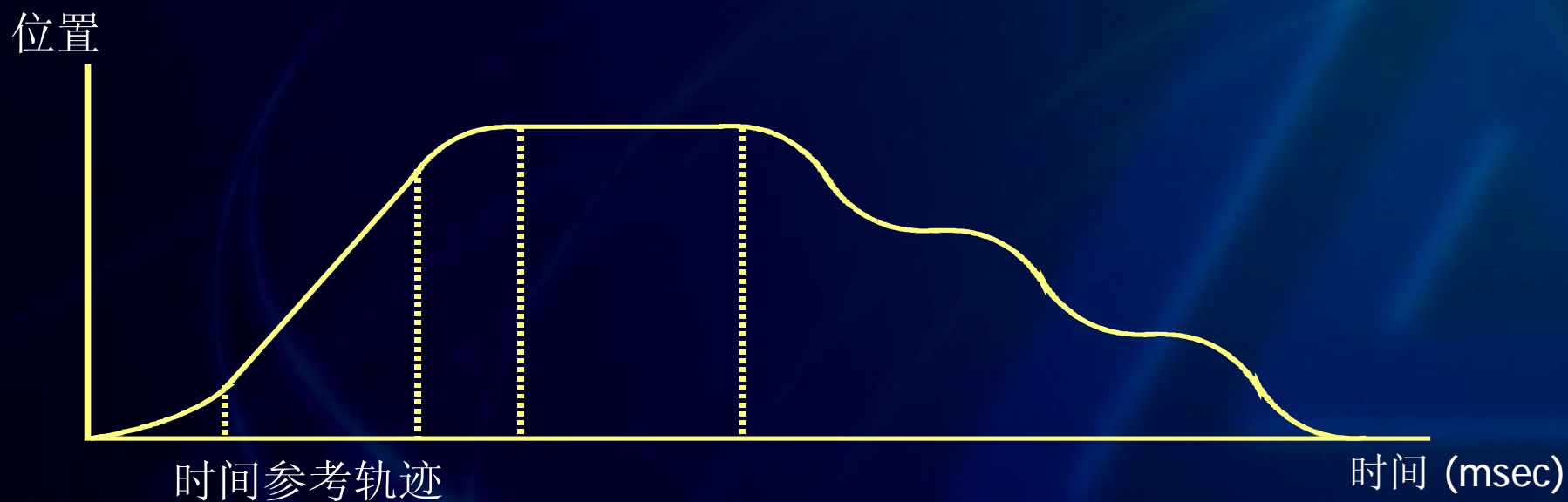
什么是外部时基?

- n 在PMAC的运动语言中，将命令位置描述为时间的函数：
 $CP=f(t)$
- n PMAC的时间基本单位为毫秒
- n 在从动的应用中，需要将主命令位置描述为主控位置的函数：
 $CP=f(MP)$

什么是外部时基？ (续)

- n 由于程序语言是以时间为单位，因此必须定义“时间” (单位: msec)和主控位置 (单位: cts)之间的关系
- n 实时输入频率 (RTIF)的单位是cts/msec
- n 如果使用RTIF为32 cts/msec，则表示在以毫秒为单位的程序中，主控位置的32个计数表示1毫秒

外部时基概念



外部时基怎样工作？

- n 转换表计算

$$\Delta MP_n = 2 * SF * (MP_n - MP_{n-1})$$

- n **Isx93** 包含转换表中 ΔMP_n 寄存器的地址

- n 坐标系中使用 ΔMP_n 作为“ Δt ”

$$CP_n = CP_{n-1} + CV_n \Delta t$$

- n 轨迹为主位置的函数

PMAC外部时基示例

动态切削：从动于web的旋转切割

Web 编码盘：5 英寸的周长

$$500 \text{ lines/转} = 2000 \text{ cts/转} \text{ (often power of 2)}$$

最大web速度= 250 in/sec

$$250 \frac{\text{in}}{\text{sec}} \cdot \frac{\text{rev}}{5\text{in}} \cdot 2000 \frac{\text{cts}}{\text{rev}} \cdot \frac{\text{sec}}{1000 \text{ msec}} = 100 \frac{\text{cts}}{\text{msec}}$$

定义“实时输入频率”(RTIF)

$$\text{RTIF} = 128 \text{ cts/msec (usually power of 2)}$$

计算“时基比例因子 (TBSF)

$$\text{TBSF} = \frac{2^{17}}{\text{RTIF} \left(\frac{\text{cts}}{\text{msec}} \right)} = \frac{131,072}{128} = 1024 \text{ (must be integer)}$$

PMAC外部时基示例(续)

在RTIF, web速度为:

$$250 \frac{\text{in}}{\text{sec}} \bullet \frac{128}{100} = 320 \frac{\text{in}}{\text{sec}} \text{ or } 3.125 \frac{\text{msec}}{\text{in}}$$

假设以web速度 去编写 刀具程序:

为了用10英寸周长的刀具匹配 web 速度:

$$320 \frac{\text{in}}{\text{sec}} \bullet \frac{\text{rev}}{10\text{in}} = 32 \frac{\text{rev}}{\text{sec}}$$

切11英寸的块件, 刀具周期必须为:

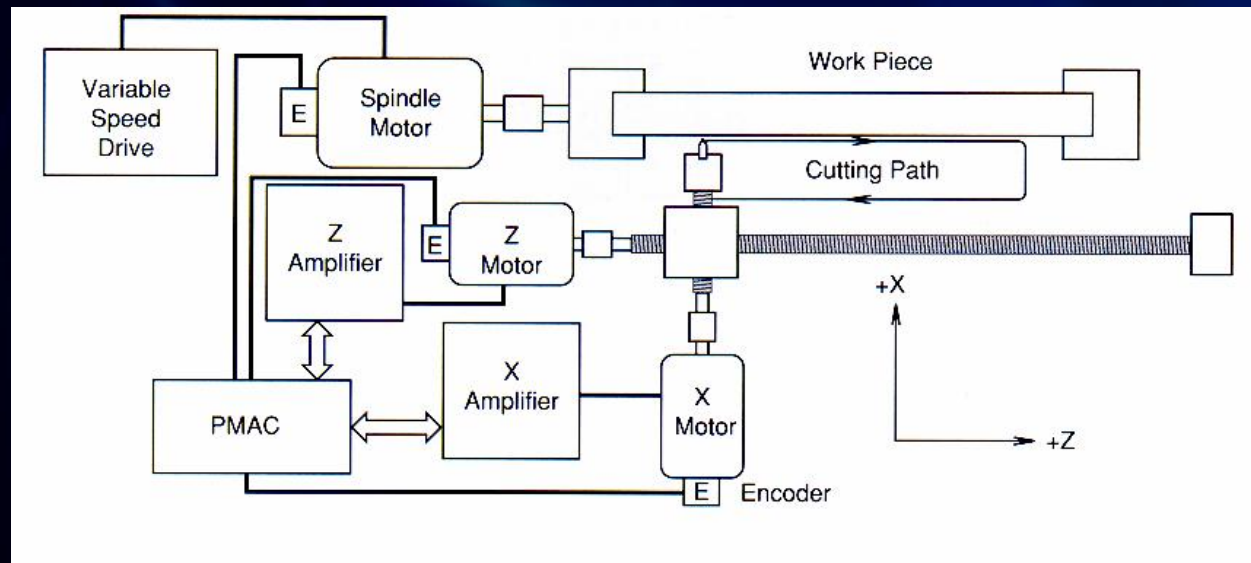
$$3.125 \frac{\text{msec}}{\text{inch}} \bullet \frac{11\text{in}}{\text{cycle}} = 34.375 \frac{\text{msec}}{\text{cycle}}$$

实例：精确从动

此例显示PMAC如何利用外部时基源进行复杂的精确从动。这里，它用于在具有开环主轴的车床上加工多线螺纹。

线圈绕制也具有一定的相似性，因为它是在加工中来传送和挤压金属的。PMAC将它的切削轴(X&Z)从动于主轴编码器，使得刀具速度跟踪主轴速度，从而得到恒定的螺距，这可以在时基模式下简单地完成。

为主轴定义一个“实时”速度，并为此设置时基常数。在假设主轴运行于实时速度的前提下为从动轴编写程序。时基控制将会自动补偿主轴速度的变换。



精确从动(续)

.***** 设置和定义 *****

,
&1 ; 定义坐标系1中的轴
#1->10000X ; 电机1在X轴方向以10000cts径向运动
#2->10000Z ; 电机2在Z轴方向以10000cts切削

;主轴编码器为1024lines/转或4096 cts/转。在实时速度3000rpm(50rps)时，编码器频率为204.8cts/msec。按公式，时基常数为 $131,072/204.8 = 640$ 。

WY:1833,640 ; 设置时基常数(在地址1833)
I193=1833 ; 告知坐标系1用此地址
; 作为时基源

.***** 运动程序内容 *****

,
这一程序用来加工一个5螺距 (每英寸5线) 螺杆。假设主轴速度为3000rpm或50rps。因此切削速度为 $50(\text{转/sec})/5(\text{转/in})=10 \text{ in/sec}$ 。为了使刀具与每道螺纹配准($50\text{转/sec} \leq 20\text{msec/转}$)，每个循环的程序时间必须精确地为20msec的倍数。

精确从动(续)

OPEN PROG 77 CLEAR	; 准备进入缓存77
P100=3.00	; X (径向)与毛坯距离
P101=P100	; 切削起始位置
RAPID X(P100-0.1) Z2	; 快速运动到起始位置
LINEAR	
WHILE (P101<3.10)	; 循环直到深度为0.1英寸
P101=P101+0.01	; 增加切深0.01英寸
TM100	; 100毫秒的“切入”时间
X(P101)	; 进入毛坯的轴半径 (“切入”)
TM(24*1000/10)	; 以10in/sec运动24英寸 (以毫秒)
Z26	; 制造24英寸螺纹 (26-2)
TM60	; 60毫秒的退刀时间
X(P100-0.1)	; 刚好退到毛坯外
TM(24*1000/30)	; 以30 in/sec运动24英寸 (以毫秒)
Z2	; 从下一个螺纹反向运动
	; 循环中运动的总时间为
	; 100+2400+60+800=3360 msec
	; 正好是20毫秒的倍数, 或主轴转一圈
ENDWHILE	; 循环结束
RAPID X0 Z0	; 返回到位置
CLOSE	

时基实时输入频率的选择约束

选择“主动计数每毫秒”来选取 RTIF 相当自由。 这里只有一些约束：

1) 为了防止数值漂移，时基比例因子 ($2^{17}/RTIF$) 必须为整数

a) $RTIF = 100 \text{ cts/msec} \Rightarrow TBSF = 1310.72 \hat{U}$ (BAD)

b) $RTIF = 128 \text{ cts/msec} \Rightarrow TBSF = 1024 \dot{U}$ (Good)

2)

$$\frac{\text{最大输入频率}}{\text{实时输入频率}} < \text{伺服更新频率} \quad (\text{kHz})$$

为了防止饱和

$$\therefore RTIF > \frac{MIF}{SUF \text{ (kHz)}}$$

a) $SUF = 2.25 \text{ kHz}, MIF = 100 \frac{\text{cts}}{\text{m sec}}, RTIF = 32 \frac{\text{cts}}{\text{m sec}} \Rightarrow RTIF < \frac{MIF}{SUF} \hat{U}$ (BAD)

b) $SUF = 2.25 \text{ kHz}, MIF = 100 \frac{\text{cts}}{\text{m sec}}, RTIF = 64 \frac{\text{cts}}{\text{m sec}} \Rightarrow RTIF > \frac{MIF}{SUF} \dot{U}$ (Good)

术语表：

RTIF: 实时输入频率(cts/msec)

TBSF: 时基比例因子

MIF: 最大输入频率(cts/msec)

SUF: 伺服更新频率

时基 I-变量 (续)

n Isx94 时基 摆率 / 摆率限制

- n 决定最大时基变化率
- n 以“ (I10的单位)/伺服周期”的形式表示
- n 设置较低的主机命令的时基 (缺省值为1644) 从而避免 step 速度改变
- n Set high for 外部时基 to prevent loss of 同步

n Isx95 进给保持摆率e

- n 同Ix94一样具有相同的单位
- n 只影响‘H’ 命令的减速和跟踪 ‘R’的再加速
- n ‘Hold’ 相当于 ‘%0’ 命令
- n Isx95 允许保持控制即使Isx94 采用 外部时基

触发时基

- n 非触发时基不是很适合作为开始主位置参考
- n 触发时基采用硬件-捕获主位置作为起始参考
- n 后续运动参考这一起始主位置

设置触发时基

- n 转换表入口
 - n 方法 \$A0 (运行触发)
 - n DSPGATE中的通道地址(e.g. \$78000)
 - n 像非触发时基一样设置比例因子
- n 触发控制
 - n 编码捕获控制 (I7m02)
 - n 捕获标志选择 (I7m03)
- n Isx93 包含对应设置内容的第二行的 地址
- n 用M-变量定义表中对应的方法字节

完成触发时基

1. 运动程序 冻结 时基和计算 第一步运动

DWELL 20	; 停止运动和前瞻
M99 = \$90	; 冻结 时基
X10	; 计算第一步移动

2. PLC程序 arms (激活) 时基

IF (M99 = \$90)	; 如果冻结
M99 = \$B0	; <u>then arm</u>
ENDIF	

3. 使用触发让转换表解除时基

- 捕获位置为起始位置(时间)
- 方法控制字被改变成\$A0 (运行)

PMAC的维护

PMAC的维护

这一章节的目的是让你获得在系统故障时如何修复的知识和工具。
假设使用PMAC的系统本来正常的工作，现在停止运行了，你需要将这一问题源隔离，使系统重新运行。

你将学习到：

这一章节将会讲解：怎样排查PMAC和一些连接的硬件以隔离问题？
如果PMAC是问题源的话，怎样解决问题？

PMAC的故障诊断

- n 任何系统中，故障诊断的关键是分割和解决。这意味着将问题分割成很小的部分，这样更容易找到原因。
- n PMAC系统中，需要考虑3个子系统。



PMAC的故障诊断

n 分隔和解决具有以下技术:

- n 1. 使用一个已知“有效的”工作配置。\$\$\$***, 使用演示软件, 等。
- n 2. 简化问题。确保没有其它的程序、特性, 或者硬件会影响状态。
- n 3. 比较。尝试其它的PMAC, 电缆, 等。

软件的故障诊断

- n 通过命令(<CTRL-D> 和 <CTRL-A>)停止所有的程序，清除可能的程序故障
- n 软件应该反映输入状态，输出应该反映指令。
- n 还原已知的工作配置或\$\$\$***

软件的故障诊断(续)

n 使用执行程序终端

- n 全局状态
- n 电机状态
- n 接头状态
- n 坐标系状态
- n 程序状态
- n 数据采集

Coordinate System Status: Device # 1 [PMAC2 Turbo] V1.945 07/02/2008: Ethernet Port

C.S.:

X:\$2040	Description	Y:\$203F	Description	Y:\$2040	Description
(bit 23)	Z-axis used in feedrate	(bit 23)	Lookahead in progress	(bit 23)	LHB wrap
(bit 22)	Z-axis increment mode	(bit 22)	Run time error	(bit 22)	(Internal)
(bit 21)	Y-axis used in feedrate	(bit 21)	(Internal)Move in stack	(bit 21)	(Internal)
(bit 20)	Y-axis increment mode	(bit 20)	Amp fault error	(bit 20)	(Internal)
(bit 19)	X-axis used in feedrate	(bit 19)	Fatal following err	(bit 19)	LHB sync M-variable overflow
(bit 18)	X-axis increment mode	(bit 18)	Warning following err	(bit 18)	LHB direction
(bit 17)	W-axis used in feedrate	(bit 17)	In position	(bit 17)	LHB stop
(bit 16)	W-axis increment mode	(bit 16)	Rotary buffer full	(bit 16)	LHB change
(bit 15)	V-axis used in feedrate	(bit 15)	Delayed calculation flag	(bit 15)	LHB last segment
(bit 14)	V-axis increment mode	(bit 14)	End of block (/) stop in progress	(bit 14)	LHB recalculate
(bit 13)	U-axis used in feedrate	(bit 13)	(Internal)Sync M-variable one-shot	(bit 13)	LHB flush
(bit 12)	U-axis increment mode	(bit 12)	(Internal)Dwell move buffered	(bit 12)	LHB last move
(bit 11)	C-axis used in feedrate	(bit 11)	Ctrr comp outside corner	(bit 11)	LHB single-segment request
(bit 10)	C-axis increment mode	(bit 10)	Ctrr comp stop req	(bit 10)	LHB change request
(bit 9)	B-axis used in feedrate	(bit 9)	Ctrr comp move buffered	(bit 9)	LHB move request

Global Status: Device # 1 [PMAC2 Turbo] V1.945 ...

X:\$6	Description	Y:\$6	Description
(bit 23)	Main error	(bit 23)	Turbo Ultralite
(bit 22)	RTI re-entry (error)	(bit 22)	Turbo VME
(bit 21)	CPU type 1	(bit 21)	CPU type
(bit 20)	Servo error	(bit 20)	Binary rotary buffer open
(bit 19)	Data gathering enabled	(bit 19)	Motion buffer open
(bit 18)	(Reserved)	(bit 18)	ASCII rotary buffer open
(bit 17)	Gather on external trig	(bit 17)	PLC buffer open
(bit 16)	Small memory Turbo PMAC	(bit 16)	UMAC Turbo
(bit 15)	(Internal)	(bit 15)	(Internal)
(bit 14)	Compensate table on	(bit 14)	(Internal)
(bit 13)	General checksum error	(bit 13)	(Reserved)
(bit 12)	Firmware checksum error	(bit 12)	(Reserved)
(bit 11)	DPRAM error	(bit 11)	Fixed buffer full
(bit 10)	EAROM error	(bit 10)	MACRO ring test enable
(bit 9)	Real time interrupt warning	(bit 9)	Ring active
(bit 8)	Illegal L-variable definition	(bit 8)	Modbus active
(bit 7)	Servo/Macro IC config. error	(bit 7)	(Reserved)
(bit 6)	TWS variable parity error	(bit 6)	(Reserved)
(bit 5)	MACRO communication error	(bit 5)	MACRO ring rcvd break msg
(bit 4)	MACRO ring error	(bit 4)	MACRO ring break
(bit 3)	Phase clock missing	(bit 3)	MACRO ring synch packet fault
(bit 2)	(Reserved)	(bit 2)	(Reserved)
(bit 1)	All cards addressed serially	(bit 1)	(Reserved)
(bit 0)	This card addressed serially	(bit 0)	Abort Input

Motor Status: Device # 1 [PMAC2 Turbo] V1.945 ...

Motor:

X:\$80	Description	Y:\$C0	Description
(bit 23)	Motor activated (lxx00)	(bit 23)	(CS-1) # bit 3 (MSB)
(bit 22)	Negative end limit set (soft or hard)	(bit 22)	(CS-1) # bit 2
(bit 21)	Positive end limit set (soft or hard)	(bit 21)	(CS-1) # bit 1
(bit 20)	Ext servo algo ena (lxx00/lxx50)	(bit 20)	(CS-1) # bit 0 (LSB)
(bit 19)	Amplifier enabled	(bit 19)	CS Axis definition bit 3
(bit 18)	Open loop mode	(bit 18)	CS Axis definition bit 2
(bit 17)	Move timer active	(bit 17)	CS Axis definition bit 1
(bit 16)	Integration mode (lxx34;0 always)	(bit 16)	CS Axis definition bit 0
(bit 15)	Dwell in progress	(bit 15)	Assigned to C.S.
(bit 14)	Data block error	(bit 14)	(Reserved for future use)
(bit 13)	Desired velocity 0	(bit 13)	Foreground in-position
(bit 12)	Abort deceleration in progress	(bit 12)	Desired position limit stop
(bit 11)	Block request	(bit 11)	Stopped on position limit
(bit 10)	Home search in progress	(bit 10)	Home complete
(bit 9)	User-written phase ena (lxx59 bit 1)	(bit 9)	Motor Phase Request
(bit 8)	User-written servo ena (lxx59 bit 0)	(bit 8)	Phasing search error
(bit 7)	Y-addr commute enc (lxx01 bit 1)	(bit 7)	Trigger move
(bit 6)	Commutation enable (lxx01 bit 0)	(bit 6)	Integrated fatal following error
(bit 5)	Pos follow offset mode (lxx06 bit 1)	(bit 5)	I2T Amplifier fault error
(bit 4)	Pos follow ena (lxx06 bit 0)	(bit 4)	Backlash direction flag
(bit 3)	Capture on error ena (lxx97 bit 1)	(bit 3)	Amplifier fault error
(bit 2)	Software capture ena (lxx97 bit 0)	(bit 2)	Fatal following error exceeded
(bit 1)	Sign/magnitude servo ena (lxx96)	(bit 1)	Warning following error exceeded
(bit 0)	Rapid max velocity select (lxx90)	(bit 0)	In-position true

硬件的故障诊断

- n 尽可能的减少硬件连接，清除可能的硬件冲突，使用“已知”有效的配置。
 - n 利用对比：尝试其它的PMAC、电缆...
 - n 核实使用了电源
 - n 在“接头状态”窗口是否看到输入？
 - n 电压表检测输入和输出
- n 如果你按照以上两张幻灯片的步骤进行了检测，那么你应该知道故障是PMAC内部还是外部的。

故障诊断示例

- n 无位置反馈?
- n 电机不运行
- n 输入/输出不工作
- n 看门狗定时器
- n 不能与PMAC通讯

无位置反馈

- n 编码器是否有电(+5V和GND)?
- n 检查编码器解码参数 I7mn0、I900(Pmac1)、I9n1(Pmac2)
- n 检查编码转换表的设置
- n 检查变量Ixx03 (和Ixx04), 进行正确地定义
- n 电机是否激活? (I100=1)
- n 单端还是差分?
- n 用示波器检查到UMAC的编码器输入

电机不运动

- n 检查电机输出方式
- n 检查致命跟随误差
- n 检查DAC的±15V电源
- n 检查限位状态
 - n 如果限位被激活，电机将不会运动
 - n 如果需要，改变lxx24(turbo)、lx25
- n 检查lxx25，是否伺服器使能指向了正确的内存位置
- n 检查lxx02，是否输出命令指向了正确的位置
- n 检查PID和/或电机参数
 - n 在缺省值下，电机不会运动
- n 检查进给重载(% 命令)

机床IO的故障诊断

n 输入不工作

- n 确保控制寄存器的设置正确
- n 在PMAC内存位置直接检查输入
- n 确保IO板被正确地定址
- n 检查IO卡的实际输入状态
- n 检查卡的电源

n 输出不工作

- n 确保控制寄存器的设置正确
- n 直接写到输出PMAC内存位置
- n 确保IO板被正确地定址
- n 检查卡的电源

看门狗

- n 看门狗上电立即发生
 - n 检查5V电压源
 - n 没有足够的后台时间
 - n 重新初始化PMAC到出厂默认值 (E3)，下载工作源代码或者PMAC备份配置
- n 看门狗随机生成
 - n 检查5V电压源
 - n 检查看门狗定时器、计数器 (M7000->Y:\$000025(turbo))
 - n 一次运行一个PLC's

通讯

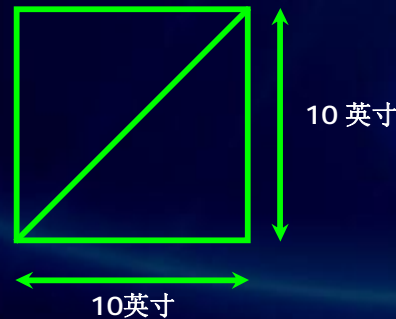
- n 即插即用(PCI)或者非即插即用(ISA)
- n 串行通讯
 - n 检查跳线(PMAC1通用cpu) 和com端口波特率的设置
 - n 强迫所有窗口和对应的卡通讯并测试(Force all window and test....)
 - n 重启...
- n USB
 - n 插拔
 - n 检查设备管理器 (delta tau usb2.0),
- n Ethernet
 - n 检查主机ip 地址
 - n Ping对应的地址 (ping 192.6.94.2)

维护和修理

- n 电池寿命 (每年改变)
- n 最好有备用的板卡
- n 是否是淘汰类型板卡
- n 修理的要求
 - n 退回维修授权 (RMA) 表格
 - n 时间要求
- n 技术支持热线
 - n 热线电话: (818) 717-5656

结业考试

老板要你用他的PMAC X, Y雕刻机画出以下形状。此雕刻机比较脆弱，因此你无法以超过1英寸每秒的速度移动刀头。两个雕刻机的电机都具有500脉冲每圈的编码器和 1英寸每圈的丝杆。



他希望当按下面板上的按钮时开始雕刻形状。并且，雕刻形状时所有的灯都亮，停止时所有的灯都灭。