§9程序举例

HOME

1:PMAC程序举例

例1:M-变量定义

本文件包括M-
变量的设置.这些定义对于I/O和寄存器的直接存取是很有用的.并不要求你一定要用到这些特殊的定义.大多数程序举例要用到这些定义.

```
CLOSE                                   ; 保证命令在线(联机)
M0..1023->*                             ; 清除所有存在的定义

M0->X:$0,0,24,U                         ; 伺服循环计数器

; 通用的输入输出(PMAC-PC,-Lite,-VME)
M1->Y:$FFC2,8,1                         ; 设备输出1
M2->Y:$FFC2,9,1                         ; 设备输出2
M3->Y:$FFC2,10,1                        ; 设备输出3
M4->Y:$FFC2,11,1                        ; 设备输出4
M5->Y:$FFC2,12,1                        ; 设备输出5
M6->Y:$FFC2,13,1                        ; 设备输出6
M7->Y:$FFC2,14,1                        ; 设备输出7
M8->Y:$FFC2,15,1                        ; 设备输出8
M9->Y:$FFC2,8,8,U                       ; 设备输出1-8按字节处理
M11->Y:$FFC2,0,1                        ; 设备输入1
M12->Y:$FFC2,1,1                        ; 设备输入2
M13->Y:$FFC2,2,1                        ; 设备输入3
M14->Y:$FFC2,3,1                        ; 设备输入4
M15->Y:$FFC2,4,1                        ; 设备输入5
M16->Y:$FFC2,5,1                        ; 设备输入6
M17->Y:$FFC2,6,1                        ; 设备输入7
M18->Y:$FFC2,7,1                        ; 设备输入8
M19->Y:$FFC2,0,8,U                      ; 设备输入1-8按字节处理

; 控制面板输入位 (如果I2=1,能用作通用I/O)

; (这些定义对PMAC-PC,-Lite,&-VME有效)
M20->Y:$FFC0,8,1                        ; Jog Minus 输入
M21->Y:$FFC0,9,1                        ; Jog Plus 输入
M22->Y:$FFC0,10,1                       ; Prejog 输入
M23->Y:$FFC0,11,1                       ; 启动(运行)输入
M24->Y:$FFC0,12,1                       ; 步进/退出输入
M25->Y:$FFC0,13,1                       ; 停止(取消)输入
M26->Y:$FFC0,14,1                       ; 返回命令输入
M27->Y:$FFC0,15,1                       ; 反馈保持输入
M28->Y:$FFC0,16,1                       ; 电机/C.S选择输入位0
M29->Y:$FFC0,17,1                       ; 电机/C.S选择输入位1
M30->Y:$FFC0,18,1                       ; 电机/C.S选择输入位2
M31->Y:$FFC0,19,1                       ; 电机/C.S选择输入位3
M32->Y:$FFC0,16,4,C                     ;  选择电机/C.S数

; 多路拨码开关接口位(能用于通用的I/O)

; (这些定义对PMAC-PC,-Lite,&-VME无效)
M40->Y:$FFC1,8,1                        ; SEL0输出
```

```
M41->Y:$FFC1,9,1                    ; SEL1输出
M42->Y:$FFC1,10,1                   ; SEL2输出
M43->Y:$FFC1,11,1                   ; SEL3输出
M44->Y:$FFC1,12,1                   ; SEL4输出
M45->Y:$FFC1,13,1                   ; SEL5输出
M46->Y:$FFC1,14,1                   ; SEL6输出
M47->Y:$FFC1,15,1                   ; SEL7输出
M48->Y:$FFC1,8,8,U                  ; SEL0-7按字节输出
M50->Y:$FFC1,0,1                    ; DAT0输入
M51->Y:$FFC1,1,1                    ; DAT1输入
M52->Y:$FFC1,2,1                    ; DAT2输入
M53->Y:$FFC1,3,1                    ; DAT3输入
M54->Y:$FFC1,4,1                    ; DAT4输入
M55->Y:$FFC1,5,1                    ; DAT5输入
M56->Y:$FFC1,6,1                    ; DAT6输入
M57->Y:$FFC1,7,1                    ; DAT7输入
M58->Y:$FFC1,0,8,U                  ; DAT0-7按字节输入

; 编码器/DAC1有关的寄存器(通常是电机#1)
M101->X:$C001,0,24,S                ; ENC1 24位位置计数器
M102->Y:$C003,8,16,S                ; DAC1 16位 模拟输出
M103->X:$C003,0,24,S                ; ENC1 捕捉/比较位置寄存器
M104->X:$C001,0,24,S                ; ENC1 位置插补(1/32ct)
M105->Y:$C006,8,16,S                ; ADC1 16位模拟输入
M106->Y:$C000,0,24,U                ; ENC1 计数间隔(SCLK周期)
M110->X:$C000,10,1                  ; ENC1 计数-写入使能控制
M111->X:$C000,11,1                  ; EQU1 比较标志位锁定控制
M112->X:$C000,12,1                  ; EQU1 比较输出使能
M113->X:$C000,13,1                  ; EQU1 比较转换使能
M114->X:$C000,14,1                  ; AENA1/DIR1输出
M116->X:$C000,16,1                  ; EQU1 比较标志
M117->X:$C000,17,1                  ; ENC1 位置-捕捉标志
M118->X:$C000,18,1                  ; ENC1 计数-出错标志
M119->X:$C000,19,1                  ; ENC1 3rd通道输入状态
M120->X:$C000,20,1                  ; HMFL1 输入状态
M121->X:$C000,21,1                  ; -LIM1 输入状态
M122->X:$C000,22,1                  ; +LIM1 输入状态
M123->X:$C000,23,1                  ; FAULT1 输入状态

; 电机#1状态位
M130->Y:$0814,11,1                  ; #1位置极限停止位
M131->X:$003D,21,1                  ; #1正极限设置位
M132->X:$003D,22,1                  ; #1反极限设置位
M133->X:$003D,13,1                  ; #1要求速度置零位
M135->X:$003D,15,1                  ; #1运行暂停位
M137->X:$003D,17,1                  ; #1运行程序
M138->X:$003D,18,1                  ; #1开环模式
M139->Y:$0814,14,1                  ; #1放大器使能状态位
M140->Y:$0814,0,1                   ; #1在位
M141->Y:$0814,1,1                   ; #1出错警告
M142->Y:$0814,2,1                   ; #1严重错误
M143->Y:$0814,3,1                   ; #1放大器出错
M145->Y:$0814,10,1                  ; #1回零完成
; 电机#1运动寄存器
M161->D:$0028                       ; #1 给定位置(1/[Ix08*32]cts)
M162->D:$002B                       ; #1 实际位置(1/[Ix08*32]cts)
M163->D:$080B                       ; #1 目标(终点)位置(1/[Ix08*32]cts)
M164->D:$0813                       ; #1 位置偏差(1/[Ix08*32]cts)
```

```
M165->L:$081F                    ; &1 X-轴目标位置(工程单位)
M166->X:$0033,0,24,S             ; #1 实际速度(1/[Ix09*32]cts/cyc)
M167->D:$002D                    ; #1 目前主控(手
                                   轮)pos(1/[Ix07*32]cts)
                                 ; 或从动电机(1/[Ix08*32]cts)
M168->X:$0045,8,16,S             ; #1 滤波器输出(DAC位)
M169->D:$0046                    ; #1 补偿校正
M170->D:$0041                    ; #1 当前相位位置；包括Y寄存器
                                   部分
M171->X:$0041,0,24,S             ; #1 当前相位位置(计数*Ix70)
M172->L:$082B                    ; #1可变微动 位置/距离(计数)
M173->Y:$0815,0,24,S             ; #1编码器捕捉初始位置偏差

; 坐标系统&1状态位
M180->X:$0818,0,1                ; &1 程序运行位
M181->Y:$0817,21,1               ; &1 圆弧-半径-错误位
M182->Y:$0817,22,1               ; &1 运行时间出错位
M184->X:$0818,4,1                ; &1 要求连续运动位
M187->Y:$0817,17,1               ; &1 在位(电机AND)
M188->Y:$0817,18,1               ; &1 出错警告位(OR)
M189->Y:$0817,19,1               ; &1 严重错误(OR)
M190->Y:$0817,20,1               ; &1 放大器出错

; 电机#1轴定义寄存器
M191->L:$0822                    ; #1 X/U/A/B/C-轴比例系数
                                   (cts/unit)
M192->L:$0823                    ; #1 Y/V-轴比例系数(cts/unit)
M193->L:$0824                    ; #1 Z/W-轴比例系数(cts/unit)
M194->L:$0825                    ; #1 轴偏移(cts)

; 坐标系统&1变量
M197->$0806,0,24,S               ; &1 主机给定的时基(I10单元)
M198->$0808,0,24,S               ; &1 当前时基(I10单元)

; 编码器/DAC2的有关寄存器(通常是电机#2)
M201->X:$C005,0,24,S             ;  ENC2 24位位置计数器
M202->Y:$C002,8,16,S             ;  DAC2 16位模拟输出
M203->X:$C007,0,24,S             ;  ENC2 捕捉/比较位置寄存器
M204->X:$0721,0,24,S             ;  ENC2 位置插补(1/32ct)
M205->Y:$C007,8,16,S             ;  ADC2 16位模拟输入
M206->Y:$C004,0,24,U             ;  ENC2 计数间隔(SCLK周期)
M210->X:$C004,10,1               ;  ENC2 计数-写入使能控制
M211->X:$C004,11,1               ;  EQU2 比较标志位锁存控制
M212->X:$C004,12,1               ;  EQU2 比较输出使能
M213->X:$C004,13,1               ;  EQU2 比较转换使能
M214->X:$C004,14,1               ;  AENA2/DIR2输出
M216->X:$C004,16,1               ;  EQU2 比较标志
M217->X:$C004,17,1               ;  ENC2 位置-捕捉标志
M218->X:$C004,18,1               ;  ENC2 计数-出错标志
M219->X:$C004,19,1               ;  ENC2  3rd通道输入状态
M220->X:$C004,20,1               ;  HMFL2 输入状态
M221->X:$C004,21,1               ;  -LIM2 输入状态
M222->X:$C004,22,1               ;  +LIM2 输入状态
M223->X:$C004,23,1               ;  FAULT2 输入状态

; 电机#2状态位
M230->Y:$08D4,11,1               ; #2位置极限停止位
M231->X:$0079,21,1               ; #2正向极限设置位
M232->X:$0078,22,1               ; #2反向极限设置位
```

```
M233->X:$0079,13,1                ; #2要求速度置零位
M235->X:$0079,15,1                ; #2运行暂停位
M237->X:$0079,17,1                ; #2运行程序
M238->X:$0079,19,1                ; #2开环模式
M239->Y:$08D4,14,1                ; #2放大器使能状态位
M240->Y:$08D4,0,1                 ; #2在位
M241->Y:$08D4,1,1                 ; #2出错警告
M242->Y:$08D4,2,1                 ; #2严重错误
M243->Y:$08D4,3,1                 ; #2放大器出错
M245->Y:$08D4,10,1                ; #2回零完成

; 电机#2运动寄存器
M261->D:$0064                     ; #2给定位置(1/[Ix08*32]cts)
M262->D:$0067                     ; #2实际位置(1/[Ix08*32]cts)
M263->D:$08CB                     ; #2目标(终点)位置(1/[Ix08*32]cts)
M264->D:$08D3                     ; #2位置偏差(1/[Ix08*32]cts)
M265->X:$0820                     ; &1 Y-轴目标位置(工程单位)
M266->D:$006F,0,24,S              ; #2实际速度(1/[Ix09*32]cts/cyc)
M267->D:$0069                     ; #2目前主控(手
                                    轮)pos(1/[Ix07*32]cts
                                  ; #2或从动电机(1/[Ix08*32]cts)
M268->X:$0081,8,16,S              ; #2滤波器输出(DAC位)
M269->D:$0082                     ; #2补偿校正
M270->D:$007D                     ; #2当前相位位置；包括Y寄存器
                                    部分
M271->X:$007D,0,24,S              ; #2当前相位位置(计数*Ix70)
M272->L:$08EB                     ; #2可变微动 位置/距离(计数)
M273->Y:$08D5,0,24,S              ; #2编码器捕捉初始位置偏差

; 坐标系&2状态位
M280->X:$08D8,0,1                 ; &2 程序运行
M281->Y:$08D7,21,1                ; &2 圆弧-半径-错误
M282->Y:$08D7,22,1                ; &2 运行时间出错
M284->X:$08D8,4,1                 ; &2 要求连续运动
M287->Y:$08D7,17,1                ; &2 在位(电机AND)
M288->Y:$08D7,18,1                ; &2 出错警告位(OR)
M289->Y:$08D7,19,1                ; &2 严重错误(OR)
M290->Y:$08D7,20,1                ; &2 放大器出错

; 电机#2轴定义寄存器
M291->L:$08E2                     ; #2 X/U/A/B/C-轴比例系数
                                    (cts/unit)
M292->L:$08E3                     ; #2 Y/V-轴比例系数(cts/unit)
M293->L:$08E4                     ; #2 Z/W-轴比例系数(cts/unit)
M294->L:$08E5                     ; #2 轴偏移(cts)

; 坐标系统&2变量
M297->$08C6,0,24,S                ; &2 主机给定时基(I10单位)
M298->$08C8,0,24,S                ; &2 当前时基(I10单位)

; 编码器/DAC3的有关寄存器(通常是电机#3)
M301->X:$C009,0,24,S              ; ENC3 24位位置计数器
M302->Y:$C00B,8,16,S              ; DAC3 16位 模拟输出
M303->X:$C00B,0,24,S              ; ENC3捕捉/比较位置寄存器
M304->X:$0722,0,24,S              ; ENC3 位置插补(1/32ct)
M305->Y:$C00E,8,16,S              ; ADC3 16位模拟输入
M306->Y:$C008,0,24,U              ; ENC3 计数间隔(SCLK周期)
M310->X:$C008,10,1                ; ENC3 计数-写入使能控制
M311->X:$C008,11,1                ; EQU3 比较标志位锁定控制
```

```
M312->X:$C008,12,1                ; EQU3 比较输出使能
M313->X:$C008,13,1                ; EQU3 比较转换使能
M314->X:$C008,14,1                ; AENA3/DIR3输出
M316->X:$C008,16,1                ; EQU3 比较标志
M317->X:$C008,17,1                ; ENC3 位置-捕捉标志
M318->X:$C008,18,1                ; ENC3 计数-出错标志
M319->X:$C008,19,1                ; ENC3 3rd通道输入状态
M320->X:$C008,20,1                ; HMFL3 输入状态
M321->X:$C008,21,1                ; -LIM3 输入状态
M322->X:$C008,22,1                ; +LIM3 输入状态
M323->X:$C008,23,1                ; FAULT3 输入状态

; 电机#3状态位
M330->Y:$0994,11,1                ; #3极限位置停止位
M331->X:$00B5,21,1                ; #3正向极限设置位
M332->X:$00B5,22,1                ; #3反向极限设置位
M333->X:$00B5,13,1                ; #3要求速度置零位
M335->X:$00B5,15,1                ; #3运行暂停位
M337->X:$00B5,17,1                ; #3运行程序
M338->X:$00B5,18,1                ; #3开环模式
M339->Y:$0994,14,1                ; #3放大器使能状态位
M340->Y:$0994,0,1                 ; #3在位
M341->Y:$0994,1,1                 ; #3出错警告
M342->Y:$0994,2,1                 ; #3严重错误
M343->Y:$0994,3,1                 ; #3放大器出错
M345->Y:$0994,10,1                ; #3回零完成

; 电机#3运动寄存器
M361->D:$00A0                     ; #3 给定位置(1/[Ix08*32]cts)
M362->D:$00A3                     ; #3 实际位置(1/[Ix08*32]cts)
M363->D:$098B                     ; #3 目标(终点)位置(1/[Ix08*32]cts)
M364->D:$0993                     ; #3 位置偏差 (1/[Ix08*32]cts)
M365->L:$0821                     ; &1 Z-轴目标位置(工程单元)
M366->X:$00AB,0,24,S              ; #3 实际速度(1/[Ix09*32]cts/cyc)
M367->D:$00A5                     ; #3 目前主控(手
                                    轮)pos(1/[Ix07*32]cts)
                                  ; 或从动电机(1/[Ix08*32]cts)
M368->X:$00BD,8,16,S              ; #3 滤波器输出(DAC位)
M369->D:$00BE                     ; #3 补偿校正
M370->D:$00B9                     ; #3 当前相位位置；包括Y寄存器
                                    部分
M371->X:$00B9,0,24,S              ; #3 当前相位位置(计数*Ix70)
M372->L:$09AB                     ; #3可变微动 位置/距离(计数)
M373->Y:$0995,0,24,S              ; #3编码器捕捉初始位置偏差

; 坐标系&3状态位
M380->X:$0998,0,1                 ; &3 程序运行
M381->Y:$0997,21,1                ; &3 圆弧-半径-错误
M382->Y:$0997,22,1                ; &3 运行时间出错
M384->X:$0998,4,1                 ; &3 要求连续运动
M387->Y:$0997,17,1                ; &3 在位(电机AND)
M388->Y:$0997,18,1                ; &3 出错警告位(OR)
M389->Y:$0997,19,1                ; &3 严重错误(OR)
M390->Y:$0997,20,1                ; &3 放大器出错

; 电机#3轴定义寄存器M
391->L:$09A2                      ; #3 X/U/A/B/C-
轴比例系数(cts/unit)
```

```
M392->L:$09A3                    ；#3 Y/V-轴比例系数(cts/unit)
M393->L:$09A4                    ；#3 Z/W-轴比例系数(cts/unit)
M394->L:$09A5                    ；#3 轴偏移(cts)

；坐标系统&3变量
M397->$0986,0,24,S               ；&3 主机给定时基(I10单位)
M398->$0988,0,24,S               ；&3 当前时基(I10单位)

；编码器/DAC4的有关的寄存器(通常为电机#4)
M401->X:$C00D,0,24,S             ； ENC4 24位位置计数器
M402->Y:$C00A,8,16,S             ； DAC4 16位模拟输出
M403->X:$C00F,0,24,S             ； ENC4 捕捉/比较位置寄存器
M404->X:$0723,0,24,S             ； ENC4 位置插补(1/32ct)
M405->Y:$C00F,8,16,S             ； ADC4 16位模拟输入
M406->Y:$C00C,0,24,U             ； ENC4 计数间隔(SCLK周期)
M410->X:$C00C,10,1               ； ENC4 计数-写入使能控制
M411->X:$C00C,11,1               ； EQU4 比较标志位锁定控制
M412->X:$C00C,12,1               ； EQU4 比较输出使能
M413->X:$C00C,13,1               ； EQU4 比较转换使能
M414->X:$C00C,14,1               ； AENA4/DIR4输出
M416->X:$C00C,16,1               ； EQU4 比较标志
M417->X:$C00C,17,1               ； ENC4 位置-捕捉标志
M418->X:$C00C,18,1               ； ENC4 计数-出错标志
M419->X:$C00C,19,1               ； ENC4  3rd通道输入状态
M420->X:$C00C,20,1               ； HMFL4 输入状态
M421->X:$C00C,21,1               ； -LIM4 输入状态
M422->X:$C00C,22,1               ； +LIM4 输入状态
M423->X:$C00C,23,1               ； FAULT4 输入状态

；电机#4状态位
M430->Y:$0A54,11,1               ；#4极限位置停止位
M431->X:$00F1,21,1               ；#4正向极限设置位
M432->X:$00F1,22,1               ；#4反向极限设置位
M433->X:$00F1,13,1               ；#4要求速度置零位
M435->X:$00F1,15,1               ；#4运行暂停位
M437->X:$00F1,17,1               ；#4运行程序
M438->X:$00F1,19,1               ；#4开环模式
M439->Y:$0A54,14,1               ；#4放大器使能状态位
M440->Y:$0A54,0,1                ；#4在位
M441->Y:$0A54,1,1                ；#4出错警告
M442->Y:$0A54,2,1                ；#4严重错误
M443->Y:$0A54,3,1                ；#4放大器出错
M445->Y:$0A54,10,1               ；#4回零完成

；电机#4运动寄存器
M461->D:$00DC                    ；#4给定位置(1/[Ix08*32]cts)
M462->D:$00DF                    ；#4实际位置(1/[Ix08*32]cts)
M463->D:$0A4B                    ；#4目标(终点)位置(1/[Ix08*32]cts)
M464->D:$0A53                    ；#4位置偏差(1/[Ix08*32]cts)
M465->X:$0819                    ；&1 A-轴目标位置(工程单元)
M466->D:$00E7,0,24,S             ；#4实际速度(1/[Ix09*32]cts/cyc)
M467->D:$00E1                    ；#4目前主控(手
                                   轮)pos(1/[Ix07*32]cts
                                 ；#4或从动电机(1/[Ix08*32]cts)
M468->X:$00F9,8,16,S             ；#4滤波器输出(DAC位)
M469->D:$00FA                    ；#4补偿校正
M470->D:$00F5                    ；#4当前相位位置；包括Y寄存器
                                   部分
```

```
M471->X:$00F5,0,24,S                    ; #4当前相位位置(计数*Ix70)
M472->L:$0A6B                           ; #4可变微动 位置/距离(计数)
M473->Y:$0A55,0,24,S                    ; #4编码器捕捉初始位置偏差

; 坐标系统&4状态位
M480->X:$0A58,0,1                       ; &4 程序运行
M481->Y:$0A57,21,1                      ; &4 圆弧-半径-错误
M482->Y:$0A57,22,1                      ; &4 运行时间出错
M484->X:$0A58,4,1                       ; &4 要求连续运动
M487->Y:$0A57,17,1                      ; &4 在位(电机AND)
M488->Y:$0A57,18,1                      ; &4 出错警告位(OR)
M489->Y:$0A57,19,1                      ; &4 严重错误(OR)
M490->Y:$0A57,20,1                      ; &4 放大器出错

; 电机#4轴定义寄存器
M491->L:$0A62                           ; #4 X/U/A/B/C-
轴比例系数(cts/unit)
M492->L:$0A63                           ; #4 Y/V-轴比例系数(cts/unit)
M493->L:$0A64                           ; #4 Z/W-轴比例系数(cts/unit)
M494->L:$0A65                           ; #4 轴偏移系数(cts)

; 坐标系统&4变量
M497->$0A46,0,24,S                      ; &4 用户命令设定时基(I10单元)
M498->$0A48,0,24,S                      ; &4 当前时基(I10单元)

; 编码器/DAC5的有关寄存器(通常是电机#5)
M501->X:$C011,0,24,S                    ; ENC5 24位位置计数器
M502->Y:$C013,8,16,S                    ; DAC5 16位 模拟输出
M503->X:$C013,0,24,S                    ; ENC5 捕捉/比较位置寄存器
M504->X:$0724,0,24,S                    ; ENC5 位置插补(1/32ct)
M505->Y:$C016,8,16,S                    ; ADC5 16位模拟输入
M506->Y:$C010,0,24,U                    ; ENC5 计数间隔(SCLK周期)
M510->X:$C010,10,1                      ; ENC5 计数-写入使能控制
M511->X:$C010,11,1                      ; EQU5 比较标志位锁存控制
M512->X:$C010,12,1                      ; EQU5 比较输出使能
M513->X:$C010,13,1                      ; EQU5 比较转换使能
M514->X:$C010,14,1                      ; AENA5/DIR5输出
M516->X:$C010,16,1                      ; EQU5 比较标志
M517->X:$C010,17,1                      ; ENC5 位置-捕捉标志
M518->X:$C010,18,1                      ; ENC5 计数-出错标志
M519->X:$C010,19,1                      ; ENC5 3rd通道输入状态
M520->X:$C010,20,1                      ; HMFL5 输入状态
M521->X:$C010,21,1                      ; -LIM5 输入状态
M522->X:$C010,22,1                      ; +LIM5 输入状态
M523->X:$C010,23,1                      ; FAULT5 输入状态

; 电机#5状态位
M530->Y:$0B14,11,1                      ; #5极限位置停止位
M531->X:$012D,21,1                      ; #5正向极限设置位
M532->X:$012D,22,1                      ; #5反向极限设置位
M533->X:$012D,13,1                      ; #5要求速度置零位
M535->X:$012D,15,1                      ; #5运行暂停位
M537->X:$012D,17,1                      ; #5运行程序
M538->X:$012D,18,1                      ; #5开环模式
M539->Y:$0B14,14,1                      ; #5放大器使能状态位
M540->Y:$0B14,0,1                       ; #5在位
M541->Y:$0B14,1,1                       ; #5出错警告
M542->Y:$0B14,2,1                       ; #5严重错误
```

```
M543->Y:$0B14,3,1                    ；#5放大器出错
M545->Y:$0B14,10,1                   ；#5回零完成

；电机#5运动寄存器
M561->D:$0118                        ；#5 给定位置(1/[Ix08*32]cts)
M562->D:$011B                        ；#5 实际位置(1/[Ix08*32]cts)
M563->D:$0B0B                        ；#5 目标(终点)位置(1/[Ix08*32]cts)
M564->D:$0B13                        ；#5 位置偏差(1/[Ix08*32]cts)
M565->L:$081A                        ；&1 B-轴目标位置(工程单元)
M566->X:$0123,0,24,S                 ；#5 实际速度(1/[Ix09*32]cts/cyc)
M567->D:$011D                        ；#5 目前主控(手
                                       轮)pos(1/[Ix07*32]cts)
                                     ；#5从动电机(1/[Ix08*32]cts)
M568->X:$0135,8,16,S                 ；#5 滤波器输出(DAC位)
M569->D:$0136                        ；#5 补偿校正
M570->D:$0131                        ；#5 当前相位位置；包括Y寄存器
                                       部分
M571->X:$0131,0,24,S                 ；#5 当前相位位置(计数*Ix70)
M572->L:$0B2B                        ；#5可变微动 位置/距离(计数)
M573->Y:$0B15,0,24,S                 ；#5编码器捕捉初始位置偏差

；坐标系统&5状态位
M580->X:$0B18,0,1                    ；&5 程序运行
M581->Y:$0B17,21,1                   ；&5 圆弧-半径-错误
M582->Y:$0B17,22,1                   ；&5 运行时间出错
M584->X:$0B18,4,1                    ；&5 要求连续运动
M587->Y:$0B17,17,1                   ；&5 在位(电机AND)
M588->Y:$0B17,18,1                   ；&5 出错警告位(OR)
M589->Y:$0B17,19,1                   ；&5 严重错误(OR)
M590->Y:$0B17,20,1                   ；&5 放大器出错

；电机#5轴定义寄存器
M591->L:$0B22                        ；#5 X/U/A/B/C-轴比例系数
                                       (cts/unit)
M592->L:$0B23                        ；#5 Y/V-轴比例系数(cts/unit)
M593->L:$0B24                        ；#5 Z/W-轴比例系数(cts/unit)
M594->L:$0B25                        ；#5 轴偏移系数(cts)

；坐标系统&5变量
M597->$0B06,0,24,S                   ；&5 主机给定时基(I10单位)
M598->$0B08,0,24,S                   ；&5 当前时基(I10单位)

；编码器/DAC6的有关寄存器(通常是电机#6)
M601->X:$C015,0,24,S                 ； ENC6 24位位置计数器
M602->Y:$C012,8,16,S                 ； DAC6 16位模拟输出
M603->X:$C017,0,24,S                 ； ENC6 捕捉/比较位置寄存器
M604->X:$0725,0,24,S                 ； ENC6 位置插补(1/32ct)
M605->Y:$C017,8,16,S                 ； ADC6 16位模拟输入
M606->Y:$C014,0,24,U                 ； ENC6 计数间隔(SCLK周期)
M610->X:$C014,10,1                   ； ENC6 计数-写入使能控制
M611->X:$C014,11,1                   ； EQU6 比较标志位锁定控制
M612->X:$C014,12,1                   ； EQU6 比较输出使能
M613->X:$C014,13,1                   ； EQU6 比较转换使能
M614->X:$C014,14,1                   ； AENA6/DIR6输出
M616->X:$C014,16,1                   ； EQU6 比较标志
M617->X:$C014,17,1                   ； ENC6 位置-捕捉标志
M618->X:$C014,18,1                   ； ENC6 计数-出错标志
M619->X:$C014,19,1                   ； ENC6  3rd通道输入状态
M620->X:$C014,20,1                   ； HMFL6 输入状态
```

```
M621->X:$C014,21,1                ;  -LIM6 输入状态
M622->X:$C014,22,1                ;  +LIM6 输入状态
M623->X:$C014,23,1                ;  FAULT6 输入状态

;  电机#6状态位
M630->Y:$0BD4,11,1                ; #6极限位置停止位
M631->X:$0169,21,1                ; #6正向极限设置位
M632->X:$0168,22,1                ; #6反向极限设置位
M633->X:$0169,13,1                ; #6要求速度置零位
M635->X:$0169,15,1                ; #6运行暂停位
M637->X:$0169,17,1                ; #6运行程序
M638->X:$0169,19,1                ; #6开环模式
M639->Y:$0BD4,14,1                ; #6放大器使能状态位
M640->Y:$0BD4,0,1                 ; #6在位
M641->Y:$0BD4,1,1                 ; #6出错警告
M642->Y:$0BD4,2,1                 ; #6严重错误
M643->Y:$0BD4,3,1                 ; #6放大器出错
M645->Y:$0BD4,10,1                ; #6回零完成

;  电机#6运动寄存器
M661->D:$0154                     ; #6给定位置(1/[Ix08*32]cts)
M662->D:$0157                     ; #6实际位置(1/[Ix08*32]cts)
M663->D:$0BCB                     ; #6目标(终点)位置(1/[Ix08*32]cts)
M664->D:$0BD3                     ; #6位置偏差(1/[Ix08*32]cts)
M665->X:$081B                     ; &1 C-轴目标位置(工程单元)
M666->D:$015F,0,24,S              ; #6实际速度(1/[Ix09*32]cts/cyc)
M667->D:$0159                     ; #6目前主控(手
                                    轮)pos(1/[Ix07*32]cts
                                  ; 或从动电机(1/[Ix08*32]cts)
M668->X:$0171,8,16,S              ; #6滤波器输出(DAC位)
M669->D:$0172                     ; #6补偿校正
M670->D:$016D                     ; #6当前相位位置；包括Y寄存器
                                    部分
M671->X:$016D,0,24,S              ; #6当前相位位置(计数*Ix70)
M672->L:$0BEB                     ; #6可变微动 位置/距离(计数)
M673->Y:$0BD5,0,24,S              ; #6编码器捕捉初始位置偏差

;  坐标系&6状态位
M680->X:$0BD8,0,1                 ; &6 程序运行
M681->Y:$0BD7,21,1                ; &6 圆弧-半径-错误
M682->Y:$0BD7,22,1                ; &6 运行时间出错
M684->X:$0BD8,4,1                 ; &6 要求连续运动
M687->Y:$0BD7,17,1                ; &6 在位(电机AND)
M688->Y:$0BD7,18,1                ; &6 出错警告位(OR)
M689->Y:$0BD7,19,1                ; &6 严重错误(OR)
M690->Y:$0BD7,20,1                ; &6 放大器出错

;  电机#6轴定义寄存器
M691->L:$0BE2                     ; #6 X/U/A/B/C-轴比例系数
                                    (cts/unit)
M692->L:$0BE3                     ; #6 Y/V-轴比例系数(cts/unit)
M693->L:$0BE4                     ; #6 Z/W-轴比例系数(cts/unit)
M694->L:$0BE5                     ; #6 轴偏移(cts)

;  坐标系统&6变量
M697->$0BC6,0,24,S                ; &6 主机给定时基(I10单位)
M698->$0BC8,0,24,S                ; &6 当前时基(I10单位)

;  编码器/DAC7的有关寄存器(通常是电机#7)
```

```
M701->X:$C019,0,24,S                    ; ENC7 24位位置计数器
M702->Y:$C01B,8,16,S                    ; DAC7 16位 模拟输出
M703->X:$C01B,0,24,S                    ; ENC7 捕捉/比较位置寄存器
M704->X:$0726,0,24,S                    ; ENC7 位置插补(1/32ct)
M705->Y:$C01E,8,16,S                    ; ADC7 16位模拟输入
M706->Y:$C018,0,24,U                    ; ENC7 计数间隔(SCLK周期)
M710->X:$C018,10,1                      ; ENC7 计数-写入使能控制
M711->X:$C018,11,1                      ; EQU7 比较标志位锁定控制
M712->X:$C018,12,1                      ; EQU7 比较输出使能
M713->X:$C018,13,1                      ; EQU7 比较转换使能
M714->X:$C018,14,1                      ; AENA7/DIR7输出
M716->X:$C018,16,1                      ; EQU7 比较标志
M717->X:$C018,17,1                      ; ENC7 位置-捕捉标志
M718->X:$C018,18,1                      ; ENC7 计数-出错标志
M719->X:$C018,19,1                      ; ENC7 3rd通道输入状态
M720->X:$C018,20,1                      ; HMFL7 输入状态
M721->X:$C018,21,1                      ; -LIM7 输入状态
M722->X:$C018,22,1                      ; +LIM7 输入状态
M723->X:$C018,23,1                      ; FAULT7 输入状态

; 电机#7状态位
M730->Y:$0C94,11,1                      ; #7极限位置停止位
M731->X:$01A5,21,1                      ; #7正向极限设置位
M732->X:$01A5,22,1                      ; #7反向极限设置位
M733->X:$01A5,13,1                      ; #7要求速度置零位
M735->X:$01A5,15,1                      ; #7运行暂停位
M737->X:$01A5,17,1                      ; #7运行程序
M738->X:$01A5,18,1                      ; #7开环模式
M739->Y:$0C94,14,1                      ; #7放大器使能状态位
M740->Y:$0C94,0,1                       ; #7在位
M741->Y:$0C94,1,1                       ; #7出错警告
M742->Y:$0C94,2,1                       ; #7严重错误
M743->Y:$0C94,3,1                       ; #7放大器出错
M745->Y:$0C94,10,1                      ; #7回零完成

; 电机#7运动寄存器
M761->D:$0190                           ; #7 给定位置(1/[Ix08*32]cts)
M762->D:$0193                           ; #7 实际位置(1/[Ix08*32]cts)
M763->D:$0C8B                           ; #7
目标(终点)位置(1/[Ix08*32]cts)
M764->D:$0C93                           ; #7 位置偏差(1/[Ix08*32]cts)
M765->L:$081C                           ; &1 U-轴目标位置(工程单元)
M766->X:$019B,0,24,S                    ; #7 实际速度(1/[Ix09*32]cts/cyc)
M767->D:$0195                           ; #7 目前主控(手
                                          轮)pos(1/[Ix07*32]cts)
                                        ; 或从动电机(1/[Ix08*32]cts)
M768->X:$01AD,8,16,S                    ; #7 滤波器输出(DAC位)
M769->D:$01AE                           ; #7 补偿校正
M770->D:$01A9                           ; #7 当前相位位置；包括Y寄存器
                                          部分
M771->X:$01A9,0,24,S                    ; #7 当前相位位置(计数*Ix70)
M772->L:$01AB                           ; #7 可变微动 位置/距离(计数)
M773->Y:$0C95,0,24,S                    ; #7 编码器捕捉初始位置偏差

; 坐标系&7状态位
M780->X:$0C98,0,1                       ; &7 程序运行
M781->Y:$0C97,21,1                      ; &7 圆弧-半径-错误
M782->Y:$0C97,22,1                      ; &7 运行时间出错
```

```
M784->X:$0C98,4,1                    ; &7 要求连续运动
M787->Y:$0C97,17,1                   ; &7 在位(电机AND)
M788->Y:$0C97,18,1                   ; &7 出错警告位(OR)
M789->Y:$0C97,19,1                   ; &7 严重错误(OR)
M790->Y:$0C97,20,1                   ; &7 放大器出错

; 电机#7轴定义寄存器
M791->L:$0CA2                        ; #7 X/U/A/B/C-轴比例系数
                                       （cts/unit)
M792->L:$0CA3                        ; #7 Y/V-轴比例系数(cts/unit)
M793->L:$0CA4                        ; #7 Z/W-轴比例系数(cts/unit)
M794->L:$0CA5                        ; #7 轴偏移(cts)

; 坐标系&7变量
M797->$0C86,0,24,S                   ; &7主机给定时基(I10单位)
M798->$0C88,0,24,S                   ; &7 当前时基(I10单位)

; 编码器/DAC8的有关寄存器(一般为电机#8)
M801->X:$C01D,0,24,S                 ; ENC8 24位位置计数器
M802->Y:$C01A,8,16,S                 ; DAC8 16位模拟输出
M803->X:$C01F,0,24,S                 ; ENC8 捕捉/比较位置寄存器
M804->X:$0727,0,24,S                 ; ENC8 位置插补(1/32ct)
M805->Y:$C01F,8,16,S                 ; ADC8 16位模拟输入
M806->Y:$C01C,0,24,U                 ; ENC8 计数间隔(SCLK周期)
M810->X:$C01C,10,1                   ; ENC8 计数-写入使能控制
M811->X:$C01C,11,1                   ; EQU8 比较标志位锁存控制
M812->X:$C01C,12,1                   ; EQU8 比较输出使能
M813->X:$C01C,13,1                   ; EQU8 比较转换使能
M814->X:$C01C,14,1                   ; AENA8/DIR8输出
M816->X:$C01C,16,1                   ; EQU8 比较标志
M817->X:$C01C,17,1                   ; ENC8 位置-捕捉标志
M818->X:$C01C,18,1                   ; ENC8 计数-出错标志
M819->X:$C01C,19,1                   ; ENC8  3rd通道输入状态
M820->X:$C01C,20,1                   ; HMFL8 输入状态
M821->X:$C01C,21,1                   ; -LIM8 输入状态
M822->X:$C01C,22,1                   ; +LIM8 输入状态
M823->X:$C01C,23,1                   ; FAULT8 输入状态

; 电机#8状态位
M830->Y:$0D54,11,1                   ; #8极限位置停止位
M831->X:$01E1,21,1                   ; #8正向极限设置位
M832->X:$01E1,22,1                   ; #8反向极限设置位
M833->X:$01E1,13,1                   ; #8要求速度置零位
M835->X:$01E1,15,1                   ; #8运行暂停位
M837->X:$01E1,17,1                   ; #8运行程序
M838->X:$01E1,19,1                   ; #8开环模式
M839->Y:$0D54,14,1                   ; #8放大器使能状态位
M840->Y:$0D54,0,1                    ; #8在位
M841->Y:$0D54,1,1                    ; #8出错警告
M842->Y:$0D54,2,1                    ; #8严重错误
M843->Y:$0D54,3,1                    ; #8放大器出错
M845->Y:$0D54,10,1                   ; #8回零完成

; 电机#8运动寄存器
M861->D:$01CC                        ; #8给定位置(1/[Ix08*32]cts)
M862->D:$01CF                        ; #8实际位置(1/[Ix08*32]cts)
M863->D:$0D4B                        ; #8目标(终点)位置(1/[Ix08*32]cts)
M864->D:$0D53                        ; #8位置偏移(1/[Ix08*32]cts)
```

```
M865->X:$081D                    ；&1 V-轴目标位置(工程单元)
M866->D:$01D7,0,24,S             ；#8实际速度(1/[Ix09*32]cts/cyc)
M867->D:$01D1                    ；#8目前主控(手
                                   轮)pos(1/[Ix07*32]cts
                                 ；或从动电机(1/[Ix08*32]cts)
M868->X:$01E9,8,16,S             ；#8滤波器输出(DAC位)
M869->D:$01EA                    ；#8补偿校正
M870->D:$01E5                    ；#8当前相位位置；包括Y寄存器
                                   部分
M871->X:$01E5,0,24,S             ；#8当前相位位置(计数*Ix70)
M872->L:$0D6B                    ；#8可变微动 位置/距离(计数)
M873->Y:$0D55,0,24,S             ；#8编码器捕捉初始位置偏差

；坐标系&8状态位
M880->X:$0D58,0,1                ；&8 程序运行
M881->Y:$0D57,21,1               ；&8 圆弧-半径-错误
M882->Y:$0D57,22,1               ；&8 运行时间出错
M884->X:$0D58,4,1                ；&8 要求连续运动
M887->Y:$0D57,17,1               ；&8 在位(电机AND)
M888->Y:$0D57,18,1               ；&8 出错警告位(OR)
M889->Y:$0D57,19,1               ；&8 严重错误(OR)
M890->Y:$0D57,20,1               ；&8 放大器出错

；电机#8轴定义寄存器
M891->L:$0D62                    ；#8 X/U/A/B/C-
轴比例系数(cts/unit)
M892->L:$0D63                    ；#8 Y/V-轴比例系数(cts/unit)
M893->L:$0D64                    ；#8 Z/W-轴比例系数(cts/unit)
M894->L:$0D65                    ；#8 轴偏移(cts)

；坐标系&8变量
M897->$0D46,0,24,S               ；&8 主机给定时基(I10单位)
M898->$0D48,0,24,S               ；&8 当前时基(I10单位)

；附件14I/O M-变量(第一块 ACC-14)
M900->Y:$FFD0,0,1                ；MI/O0
M901->Y:$FFD0,1,1                ；MI/O1
M902->Y:$FFD0,2,1                ；MI/O2
M903->Y:$FFD0,3,1                ；MI/O3
M904->Y:$FFD0,4,1                ；MI/O4
M905->Y:$FFD0,5,1                ；MI/O5
M906->Y:$FFD0,6,1                ；MI/O6
M907->Y:$FFD0,7,1                ；MI/O7
M908->Y:$FFD0,8,1                ；MI/O8
M909->Y:$FFD0,9,1                ；MI/O9
M910->Y:$FFD0,10,1               ；MI/O10
M911->Y:$FFD0,11,1               ；MI/O11
M912->Y:$FFD0,12,1               ；MI/O12
M913->Y:$FFD0,13,1               ；MI/O13
M914->Y:$FFD0,14,1               ；MI/O14
M915->Y:$FFD0,15,1               ；MI/O15
M916->Y:$FFD0,16,1               ；MI/O16
M917->Y:$FFD0,17,1               ；MI/O17
M918->Y:$FFD0,18,1               ；MI/O18
M919->Y:$FFD0,19,1               ；MI/O19
M920->Y:$FFD0,20,1               ；MI/O20
M921->Y:$FFD0,21,1               ；MI/O21
M922->Y:$FFD0,22,1               ；MI/O22
```

```
M923->Y:$FFD0,23,1                    ; MI/O23
M924->Y:$FFD0,0,1                     ; MI/O24
M925->Y:$FFD0,1,1                     ; MI/O25
M926->Y:$FFD0,2,1                     ; MI/O26
M927->Y:$FFD0,3,1                     ; MI/O27
M928->Y:$FFD0,4,1                     ; MI/O28
M929->Y:$FFD0,5,1                     ; MI/O29
M930->Y:$FFD0,6,1                     ; MI/O30
M931->Y:$FFD0,7,1                     ; MI/O31
M932->Y:$FFD0,8,1                     ; MI/O32
M933->Y:$FFD0,9,1                     ; MI/O33
M934->Y:$FFD0,10,1                    ; MI/O34
M935->Y:$FFD0,11,1                    ; MI/O35
M936->Y:$FFD0,12,1                    ; MI/O36
M937->Y:$FFD0,13,1                    ; MI/O37
M938->Y:$FFD0,14,1                    ; MI/O38
M939->Y:$FFD0,15,1                    ; MI/O39
M940->Y:$FFD0,16,1                    ; MI/O40
M941->Y:$FFD0,17,1                    ; MI/O41
M942->Y:$FFD0,18,1                    ; MI/O42
M943->Y:$FFD0,19,1                    ; MI/O43
M944->Y:$FFD0,20,1                    ; MI/O44
M945->Y:$FFD0,21,1                    ; MI/O45
M946->Y:$FFD0,22,1                    ; MI/O46
M947->Y:$FFD0,23,1                    ; MI/O47
```

例2: 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.首先程序规定了怎样运动,然后执行运动.

```
; ********************************设置和定义*************************************************************

&1                                    ; 坐标系统1
CLOSE                                 ; 确保所有缓存关闭
#1->X                                 ; 电机#1定义为X轴，X轴一个
                                      ; 编程单位就是一个编码计数

; ********************************运动程序文本*************************************************************

OPEN PROG1                            ; 打开缓存,为#1电机编程输入程序
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
CLOSE                                 ; 关闭缓存,结束程序

运行此程序
&1  B1  R                             ; 坐标系1,指向程序开始,运行程序
```

例3: 较复杂的运动

本程序介绍增量运动和规定时间的运动,循环逻辑,变量应用,轴标定以及简单的算法.逻辑和数学计算没有延迟运动.

```
; ********************************设置和定义**************************************************************

&2                                          ; 坐标系统2
                                            ; 注释:一个电机不能同时定义在多个坐标系里
CLOSE                                       ; 确保所有缓存关闭
#5->1000X                                   ; 电机5的X轴一个单位(cm)计数是1000

; ********************************运动程序文本************************************************************

OPEN PROG2                                  ; 打开程序#2的缓存入口
CLEAR                                       ; 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
P1=0                                        ; 初始化循环计数变量
WHILE (P1<10)                               ; 循环到条件为假(10次)
    X10                                     ; X-轴运动10cm(=1000cts)正向
    DWELL500                                ; 停止1/2秒
    X-10                                    ; X-轴反向运动10cm
    DWELL500                                ; 停止1/2秒
    P1=P1+1                                 ; 计数加1
ENDWHILE                                    ; 循环结束
CLOSE                                       ; 关闭缓存-结束程序
```

运行此程序:

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

例4:条件分支

此例介绍了条件分支,计算运动距离寻找零点及访问I/O

```
; ******************************设置和定义**************************************************************

CLOSE                                       ; 确保所有缓存关闭
&1                                          ; 坐标系1
#2->27.77777778A                            ; A-轴按角度编程
                                            ; 10000(cts/rev)/360(deg/rev)
M1->Y:$FFC2,8,1                             ; 变量M1分配到设备输出1
M11->Y:$FFC2,0,1                            ; 变量M11分配到设备输入1
I190=60000                                  ; 回馈率单位为分
                                            ; (1分=60000毫秒)

; ********************************运动程序文本************************************************************

OPEN PROG 3 CLEAR                           ; 为进入准备缓存
HOME2                                       ; 寻找电机回零位置
LINEAR                                      ; 直线插补运动方式
F20                                         ; 速度为20度/分
Q50=0                                       ; 初始化循环计数变量
WHILE  (Q50<36)                             ; 循环直到条件为假(36次)
IF (M11=1)                                  ; 设备输出1打开否?
A((Q50+1)*10)                               ; 正向运动到计算位置
```

```
ELSE                                    ；条件为假执行下面分支
A(-(Q50+1)*10)                          ；反向运动到计算位置
ENDIF
DWELL20                                 ；停止20毫秒
M1=1 M1=0                               ；输出脉冲后快速关断
DWELL20                                 ；停止20毫秒
A0                                      ；返回初始位置
Q50=Q50+1                               ；循环计数加1
ENDWHILE                                ；循环结束
CLOSE                                   ；关闭缓存-程序结束

                                        ；运行程序

&1 B3 R                                 ；坐标系1,指向程序3入口,运行
```

例5:直线和圆弧插补

本例介绍在X-Y直角坐标系中直线和圆弧插补的应用

```
；**********************************设置和定义**********************************************************

CLOSE                                   ；确保所有缓存关闭
&1                                      ；坐标系1
#3->10000X                              ；X轴用3号电机
#4->10000Y                              ；Y轴用4号电机

；**********************************旋转轴运动文本******************************************************

OPEN PROG 4 CLEAR                       ；准备进入
RAPID X1 Y4                             ；快速运动到始点
F500                                    ；直线和圆弧运动速度
LINEAR Y13                              ；直线运动
CIRCLE1 X2 Y14 I1 J0                    ；CW圆弧
LINEAR X3                               ；直线运动
CIRCLE1 X4 Y13 I0 J-1                   ；CW圆弧
LINEAR Y7                               ；直线运动
CIRCLE1 X7 Y4 I3 J0                     ；CCW圆弧
LINEAR X13                              ；直线运动
CIRCLE1 X14 Y3 I0 J-1                   ；CW圆弧
LINEAR Y2                               ；直线运动
CIRCLE1 X13 Y1 I-1 J0                   ；CW圆弧
LINEAR X4                               ；直线运动
CIRCLE1 X1 Y4 I0 J3                     ；CW圆弧
DWELL100                                ；保持100毫秒
RAPID X0 Y0                             ；返回
CLOSE

；运行程序

&1 B4 R                                 ；坐标系1,指向程序4开始,运行
```

例6: 简单的G-代码程序

这是一个简单的PMAC G-
代码程序.Gxx的意PROG1000(下面)程序,Mxx的意思是调用标号为Nxx000的PROG1001(底部)程序.可能有更多的扩展代码；磁盘提供了一般代码的标准形式.

```
; ********************************分程序文本*************************************************************************

注释:在这部分程序中不需知到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1800rmp
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                                              ; 降低切速
X30.25 5.00                                         ; 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                                          ; 剪切上移,停止
G00 X0 Y0                                           ; 返回初始坐标
CLOSE

;          ***************************************************************************************************
***********

运动程序1000包含G-代码子程序
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线性插补方式
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2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相对方式
IF (Q100 & 262144 >0)                               ; S参数指定否?
I422=Q119/30                                        ; #4手动速度为cts/msec
ENDIF
RETURN
CLOSE

; ***************************************************************************************************************
*

运动程序1001包括M-代码子程序
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(ditto)
N05000 CWD :#4J/" RET                               ; 停止主轴
CLOSE

; 运行程序
```

```
&1 B5 R                                                    ；坐标系1,指向程序5开始运行
```

例7:机器人应用

此例说明用PMAC实现机器人控制是多么快捷和方便.此例运用选项2的双口RAM以最快速度传递数值.(若用普通的通信端口,性能略有下降)这时,PC主机每5毫秒进行一次逆运动计算,将指尖位置转换为关节位置,并将6个关节位置送入DPROM.在那里它们作为M-
变量被存储.PMAC在各轴中间点之间用三次样条进行插补,在每个伺服循环中都要被更新一次.

```
；**********************************设置和定义*************************************************************

CLOSE                                                      ；确保所有缓存关闭
&1                                                         ；在坐标系统1中以角度定义轴
#1->1024X                                                  ；1024cts/度
#2->2000Y                                                  ；2000cts/度
#3->2000Z                                                  ；2000cts/度
#4->5000A                                                  ；5000cts/度
#5->512B                                                   ；512cts/度
#6->277.77778C                                             ；277.77778cts/度
M70->F:$D200                                               ；DPROM中的32位IEEE浮点
M71->F:$D201                                               ；运算变量用来给PMAC传送关
M72->F:$D202                                               ；节位置,主机发送用指针变量来
M73->F:$D203                                               ；写入这些数据
M74->F:$D204                                               ；
M75->F:$D205                                               ；
M113->F:$C000,13,1                                         ；中断位提示主机设置下一点

**********************************运动程序文本*************************************************************

OPEN PROG 75 CLEAR
SPLINE1                                                    ；三次样条方式
TA5                                                        ；设置样条部分时间为5毫秒
INC                                                        ；增量方式
X0 Y0 Z0 A0 B0 C0                                          ；第一次0距离运动
X0 Y0 B0 A0 Z0 C0                                          ；第二次0距离运动
ABS                                                        ；进一步的运动取绝对值
WHILE(P1>0)                                                ；主机设置P1来退出此模式
X(M70) Y(M71) Z(M72) A(M73) B(M74) C(M75) M113=1 M113=0
                                                           ；计算下一部分,通过中断向主机
                                                           ；索取下一数据
ENDWHILE
CLOSE
```

例8: X-Y系统的切线旋转

此例介绍的是如何简单地将旋转轴与XY系统的运动相切(或相交).旋转轴由一个同时运行的独立程序控制,它可以监视X-Y变量,XY程序部分不必知道旋转轴的任何情况,所有都是自动进行的.

```
**********************************设置和定义*************************************************************

CLOSE
&1 #1->10000X                                              ；坐标系1的1号电机是X-轴
#2->10000Y                                                 ；坐标系1的2号电机是Y-轴
&2 #3->40C                                                 ；坐标系2的3号电机是C-轴
I15=0                                                      ；角度三角运算
I327=14400                                                 ；C轴每14400cts翻转
M161->D:$0028                                              ；电机1当前给定位置
```

```
M163->D:$080B                          ; 电机1目标位置
M261->D:$0064                          ; 电机2当前给定位置
M263->D:$08CB                          ; 电机2目标位置
P10=3072                               ; 在 Mx61 Mx63 单位计数一次(被0除检验)

**************************旋转轴运动程序文本*************************************************************

OPEN PROG   10 CLEAR
SPLINE1 TA20                           ; 20毫秒的三次样条部分时间
WHILE (P1>0)                           ; 在此模式中尽量长
Q0=M163-M161                           ; X目标位置-X目前位置
Q1=M263-M261                           ; Y目标位置-Y目前位置
IF(ABS(Q0)>P10 OR ABS(Q1)>P10)         ; 目标位置<>目前位置
Q2=ATAN2(Q1)                           ; 计算角度-ATAN(Q1,Q2)
ENDIF
C(Q2)                                  ; 使旋转轴运动到新位置
ENDWHILE
CLOSE
```

例9: PLC输入编程

此例介绍如何利用PLC程序生成特定的输入。当PLC程序不断地在后台重复循环时,它检查输入线并在合适的时候发出命令。本程序显示如何用一个拨码开关输入线生成一个特定电机专用的微动开关。注意利用锁定标志,使命令只有在状态改变时才给出,并且是边沿触发。

```
; ******************************设置和定义*********************************************************

CLOSE
M50->Y:$FFC1,0                         ; 拨码开关输入口 位0
M60->*                                 ; 锁定M50位

; ***************************PLC 程序文本*********************************************************

OPEN PLC 16
CLEAR
IF (M50=1)                             ; 电机1  jog plus打开
IF(M60=1)                              ; 不在最后时间
COMMAND " " #1J/ " "                   ; 发出停止命令
M60=0                                  ; 设置锁定标志
ENDIF
ENDIF
CLOSE
```

例10: PLC显示程序

此例显示如何利用PLC程序不断地更新寄存器以及显示当前状态导出的数值.在这里,程序从一个R-Theta表中获得电机的实际位置,在将它们转换成X-Y位置,并将它们送入双口RAM以备主机使用,并将它们送到LCD进行显示。

```
*******************************设置和定义*********************************************************

CLOSE
#1->-100U+15000                        ; 径向臂以100cts/mm速度偏移150mm
#2->40C                                ; 旋转轴以40cts/deg的速度运动
M162->D:$002B                          ; 电机1实际位置
M262->D:$0067                          ; 电机2实际位置
M100->F:$D400                          ; 返回X位置用DPRAM寄存器
M200->F:$D401                          ; 返回Y位置用DPRAM寄存器
```

**********PLC程序文本**********

```
OPEN PLC 6 CLEAR
P162=150-M162/(I108*32*100)          ; 径向距离以mm从中心计算
P262=M262/(I208*32*40)               ; 旋转位置按角度计算
M100=P162*COS(P262)                  ; X位置mm=>DPRAM寄存器
M200=P162*SIN(P262)                  ; Y位置mm=>DPRAM寄存器
DISPLAY 5,10,2,M100                  ; 按格式显示到LCD
DISPLAY 15,10,2,M200                 ; 按格式显示到LCD
CLOSE
```

例11: 精确从动

此例显示PMAC如何利用一个外部时钟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。绕制线圈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,并以此设置时基常数。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化。

**********设置和定义**********

```
&1                                   ; 在坐标系统1中定义轴
#1->10000X                           ; 电机1在X轴方向以10000cts径
                                       向运动
#1->10000Z                           ; 电机2在Z轴方向以10000cts切削

; 轴编码器为1024轨/rev或4096cts/rev.在实时速度为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)
1193=1833                            ; 通知坐标系统1用此地址作为时基
                                       来源
```

**********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.或者50rps.因此切削速度为50(rev/sec)/5(rev/in)=10in/sec.为了使刀具与每道螺纹配准(50rev/sec<=>20msec/rev),
每个循环的程序时间必须精确地为20msec的倍数.

```
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
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
  TM100                              ; 100msec " 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                               ; 60msec退刀时间
  X(P100-0.1)                        ; 刚好退到毛坯外
  TM(24*1000/30)                     ; 以30in/sec运动24英寸
  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=3360msec,
                                     ; 正好是20msec的倍数,或主轴转一圈
ENDWHILE                             ; 结束循环
RAPID X0 Z0                          ; 返回初始位置
CLOSE
```

例12:运行中的位置匹配

此例显示如何利用PMAC的位置捕捉功能在运行中与一个不受PMAC控制的运动系统相匹配。在这个特殊的例子里,有一个短传送带以不均匀时间传送物品,而在第二个变速驱动的传送带上将它们用等间隔的挡板隔开。对于任何以随机间隔输入物品而要求以等间隔输出物品的加工流程,这项技术都非常有用。

这项应用中用到了PMAC的硬件位置捕捉寄存器。一旦接收到外部的触发信号,精确的位置就被保存起来。位置捕捉与驱动速度完全无关。这使高速的记录和校正成为可能。为使位置校正变得最小,这项应用中的传送带还可以从动于带挡板的传送带,在时基模式下运行以便于速度匹配(参见例10)

**********设置和定义**********

```
CLOSE
&1 #1->X                          ; X轴是#1电机，1个计数的比例系数
M11->Y:$FFC2,0,1                  ; M11为设备输入1
M203->X:$C007,0,24,S              ; 编码器2捕捉位置寄存器
M217->X:$C004,17,1                ; 编码器2捕捉位置标志
1907=2                            ; 在标志上升沿捕捉ENC2
1908=0                            ; 用HMFL2捕捉ENC2
P1=5000                           ; 每个挡板ENC2计数5000
P2=1325                           ; 捕捉时挡板周期内的
                                  ; ”“无校正”位置
```

**********运动程序内容**********

```
OPEN PROG 346 CLEAR
INC                               ; 增量运动方式
SPLINE1 TA10                      ; 每段10msec的三次样条
WHILE (M11=1)                     ; 按每输入一次设置循环一次
  WHILE (M217=0)                  ; 循环直到光电开关被挡住
    X(P10)                        ; P10是10msec的标定距离
  ENDWHILE                        ; 循环返回再次检查触发器
                                  ; 并进行计算调整
  P3=M203%P1                      ; 读入传送带挡板的捕捉位置,用每个
                                  ; 挡板的计数值对其求模,
                                  ; 找出在挡板周期中的位置
  X(P10+P4/4)                     ; 在10msec内补偿第一个1/4误差
  X(P10+P4/2)                     ; 在10msec内补偿1/4误差
  X(P10+P4/4)                     ; 在10msec内补偿后一个1/4误差
ENDWHILE                          ; 循环返回到下一个操作
```

例13: 主轴运行

此程序提供一种精密主轴操作方法,包括表面恒定速度(CSS)模式。它把主轴的运动分为非常小的段(每种情况25msec),所以能在改变速度时作出反应。此例需要在X轴用电机#1作为CSS模式的控制尺度。在CSS模式,当刀尖在径向上的尺度变化时,轴的速度也是变化的,
这样,刀尖经过工件表面时,表面速度可以保持恒定。重要的是,轴的旋转速度和刀尖的径向进给成反比(X轴位置+/-偏移量)。因为这样会导致旋转速度无限增大,所以设定G92为最大旋转速度。

本程序需要以下变量:
M55.....标志位,指示主轴是正转(ON CW)(=+1),反转(ON CCW)(=-1),或停止(OFF)(=0)
由部分程序设置
P97.....需要的主轴速度(RPM),由部分程序设置--绝对值
P96.....需要的主轴表面速度(ft/min 或m/min).由分程序设置(本例中P96由P97计算得出)
---绝对值
P95.....主轴最大加速度(RPM/sec),设置为系统常数
P92.....主轴最大允许速度(RPM),在分程序中设置(G92S)

P93.....主轴程序每秒分段数,设置为系统常数
M96.....标志位,指示CSS模式(=1)或非CSS模式(=0),由部分程序设置
M70.....标志位,指示公制(=1)或英制(=0), 由部分程序设置
M95.....标志位,指示正向加速(=+1),反向加速(=-1)或无加速(=0),由轴程序设置
P197.....当前段的给定主轴速度(RPM)不必与需要的主轴速度P97相同.但应趋近该值
M162.....保存电机#1(X轴)实际位置的寄存器,单位是1/(I108*31)编码器计数值
M164.....保存电机#1位置偏移量的寄存器
P162.....保存电机#1位置的变量转换到计数
M191.....保存X轴位置(用户单位)和电机#1位置(计数值)比例的寄存器,
在轴定义部分说明
M194.....保存X轴0位与电机#1 0位偏移量的寄存器, 在轴定义部分说明
P98.....保存X轴0位与轴中心偏差的寄存器,在分程序中设置(G92R)
P163.....保存X轴与主轴中心距离的寄存器,第一个电机#1的位置转换为
轴位置[(P162-M194)/M191],然后减去主轴中心偏移量.

```
; ************************************设置和定义************************************************************

P1=3.14159272                                   ; PI--角度变换到表面速度
P95=1000                                        ; 主轴最大加速度(RPM)
P93=40                                          ; 每秒40段
M55->*                                          ; 定义为自参照标志
M95->*                                          ; 同上
M96->*                                          ; 同上
M70->*                                          ; 同上
M95->*                                          ; 同上
M162->D:$002B                                   ; 电机#1实际位置寄存器
                                                ; (在版本1.09或更老版本中为
                                                  D:$002C)
M164->D:$0813                                   ; 电机#1位置偏移寄存器
M191->L:$0822                                   ; 电机#1 X轴系数
M194->L:$0825                                   ; 电机#1 偏移系数

; ****************************实际主轴程序*************************************************************************

CLOSE
OPEN PROG 1010
CLEAR
LINEAR                                          ; 直线插补运动
INC(C)                                          ; 规定增量运动
TM(1000/P93)                                    ; 运动范围,以毫秒计
TA(1000/P93)                                    ; 整段加速时间
TS0                                             ; 直线加速
P197=0                                          ; 起始速度为0 RPM
WHILE (M55!=0 OR P197!=0)
  IF( M96=1)                                    ; 表面恒定速度模式?
    P162=ABS((M162+M164)/(I108*32))             ; 电机#1实际位置
    P163=(P162-M194)/M191-P98                   ; X轴径向距离
    IF(ABS(P163)>0.1)
        P97=p96/(2*p1*p163)                     ; 转换为旋转速度
        IF(M70=0)  P97=12*P97                   ; 英制换算(in->ft)
        ELSE P97=1000*P97                       ; 公制换算(mm->m)
        IF (P97>P92)                            ; 超出限制否?
           P97=P92                              ; 设置极限
        ENDIF
    ELSE
        P97=P92                                 ; 设置转速极限
    ENDIF
  ENDIF
  IF (M55*P97<P197) M95=-1                      ; 要求反向加速
```

```
  IF (M55*P97>P197) M95=1                          ; 要求正向加速

  IF(M95!=0)                                       ; 改变速度否?
        P197=P197+M95*P95/P93                      ; 转向新速度
        IF((P197-M55*P97)*M95>0)                   ; 超过要求速度否?
           P197=M55*P97                            ; 设置到要求速度
           M95=0                                   ; 不再加速标志
        ENDIF
    ENDIF
    C(P197/(60*P93))                               ; 实际增值运动阶段
ENDWKILE
DWELL50
M95=0                                              ; 清除加速标志结束&1等待
CLOSE
```

例14:限制转矩的速度运动

此程序执行一个限制转矩的速度运动。当转矩低于预定极限时,轴以恒定速度运动(在这里有寄存器DAC测量,但也可以是实际的力矩/力传感器)。如果超出极限,指定的速度将下降。

```
CLOSE DELETE GATHER TRACE
M102->Y:$C003,8,16,S                               ; DAC1寄存器
P102=3200                                          ; 转矩极限
M1=1

OPEN PROG 47
CLEAR
TA10
TS0
TM10
LINEAR
INC
P1=100                                             ; 10msec的指定距离
WHILE (M1=1)                                       ; 尽量可能长地运行于这种模式
    IF (M102<P102)                                 ; 在转矩极限内
         P2=P1                                     ; 运动指定距离
    ELSE                                           ; 超出转矩极限
         IF (P2>0)
              P2=P2-5                              ; 减少距离
         ELSE
              P2=0                                 ; 不让其反向
         ENDIF
    ENDIF
    X(P2)                                          ; 进行下一阶段的运动
  ENDWHILE
  CLOSE
```

例15:限转矩起重操作

此例显示了怎样进行一次限制转矩的起重操作。要点是以小步长持续增加起重速度,直到转矩超出指定极限。在这里,应用了DAC输出(进入转矩模式放大器)。但也可用电机电流或实际的力矩传感器作为输入。

```
CLOSE
&1
#3->2000Z                                          ; 垂直轴(mm)
#4->2000C                                          ; 旋转轴(转数)
M102->Y:$C00A,8,16,S                               ; (转矩不直接测量)
```

```
P102=4000

OPEN PROG 508 CLEAR
ABS(Z,C)                                ; 两轴以绝对方式运行
PSET C0                                 ; 指定目前旋转位置为0位
P1=-5                                   ; 快速运动到Z端位置
TA50                                    ; 快速运动加速时间
F10                                     ; 快速运动速度
Z(P1)                                   ; 执行快速运动
DWELL5                                  ; 短暂停留(任选)
TA10 TM10                               ; 在循环中进行10ms的阶段运动
P2=0                                    ; C轴位置系数
WHILE (P2<3.0 AND M102<P102)            ; 循环3次或到高转矩
   P1=P1-0.05                           ; Z轴位置增值
   P2=P2+0.1                            ; C轴位置增值
   Z(P1) C(P2)                          ; 下一步起重运动
ENDWHILE                                ; 循环返回
DWELL 10                                ; 短暂停顿(任选)
TA50                                    ; 退出运动加速
F10                                     ; 退出运动速度
Z0                                      ; 退出运动命令
CLOSE
```

例16:运动中改变终点运动

本程序允许进行变化终点的运动。The card经常朝着P1的X方向和P2的Y方向运动,但是它是以小增量运动的,所以当P1或P2改变时,它将在几个毫秒内改变方向。P1和P2可以在运行过程中通过主机来改变。

```
CLOSE
OPEN PROG 2
CLEAR
P9=100                                  ; 每次增量的矢量长度
P11=0                                   ; X起始位置
P12=0                                   ; Y起始位置
I187=10                                 ; C.S.1加速时间
I188=0                                  ; C.S.1 S-曲线时间
TM10                                    ; 增量时间,以毫秒计
WHILE (M1=1)                            ; 可用任何条件
    P21=P1-P11                          ; X方向需走距离
    P22=P2-P12                          ; Y方向需走距离
    P0=SQRT(P21*P21+P22*P22)            ; 矢量距离
    IF(P0>P9)                           ; 如果大于增值
        P0=P9/P0                        ; 距离比值
        P11=P11+P21*P0                  ; X方向增量
        P12=P12+P22*P0                  ; Y方向增量
    ELSE                                ; 距离<增值
        P11=P1                          ; 直接运动到终点
        P12=P2
  ENDIF
  X(P11) Y(P12)                         ; 运动
ENDWHILE
CLOSE
```

例17: 变速

本程序举例说明,在高速运动时,如果有外部命令,将快速响应命令,但速度改变很慢。通过把运动分为许多非常小的环节,并用WHILE循环在一个模式下连续运行,直到接收到信号位置来实现此功能。

```
CLOSE
OPEN PROG 10
CLEAR
P4=150000
P5=3000
P6=600
INC(X)                          ; X轴为增量模式
TM(P5/P6)                       ; 运动阶段时间(在此为5msec)
TA(P5/P6)                       ; 加速时间(<=TM)
P3=P4/(P6*P6)                   ; 计算距离
P1=0                            ; 初始速度为0
P2=0                            ; 初始距离为0
N10 WHILE (P0!=0)               ; P0=0是减速信号
  IF (P1<P6)                    ; 没到全速?
    P1=P1+1                     ; 增值加速
    P2=P1*P3                    ; 阶段运动距离
  ENDIF
  X(P2)                         ; 增量运动阶段
ENDWHILE
WHILE (P0=0)                    ; P0=1是加速信号
 IF(P1>0)                       ; 仍在运动吗?
   P1=P1-1                      ; 减值降速
   P2=P1*P3                     ; 阶段运动距离
 ENDIF
 X(P2)                          ; 增量运动阶段
ENDWHILE
GOTO 10                         ; P0=1是循环返回
CLOSE
```

例18:用线性运动进行圆插补

此例包含一个用混合运动进行园插补的子程序.起始地址为N10000的子程序进行所以圆弧运动的计算.如果主程序没有指定圆心,程序就转到N20000来计算中心点.对每个圆弧插补运动,主程序必须规定起点,终点,半径,方向和步长.长弧/短弧结合点或中心点也必须指出.因为这些规定是保存其值的变量,所以在运动中不变的量就不必重新规定.此例的主程序为子程序提供了全面的练习.

```
CLOSE
OPEN PROG 1
CLEAR

; 程序的主要部分用来练习子程序
TA50                            ; 应比增值时间短
TS0                             ; 最好的混合是0
F1000                           ; 回馈率规定是最简单的
X10000Y0                        ; 线性运动

; 顺时针长弧
Q1=10000 Q2=0 Q3=5000 Q4=-5000 Q5=5000
Q6=1 Q7=1 Q8=2  GOSUB 10000
; 线性运动
X5000Y0

; 顺时针短弧
Q1=5000 Q2=0 Q3=10000 Q4=5000 Q5=5000
Q6=1 Q7=0 Q8=2  GOSUB 10000

; 逆时针短弧
Q1=Q3 Q2=Q4 Q3=12500 Q4=7500 Q5=2500
Q6=0 Q7=0  GOSUB 10000
```

```
；逆时针长弧
Q1=Q3 Q2=Q4 Q3=10000 Q4=5000 Q5=2500
Q6=0 Q7=1 Q8=2  GOSUB 10000

；顺时针半圆
Q1=Q3 Q2=Q4 Q3=10000 Q4=0 Q5=2500
Q6-1 Q7=1 Q8=2  GOSUB 10000

；逆时针半圆；dist>2R
Q1=Q3 Q2=Q4 Q3=10000 Q4=-5005 Q5=2500
Q6=0 Q7=0 Q8=2  GOSUB 10000
DELAY 500
X0Y-5000
DELAY 500

；Now an multi-part circle to test off angles
Q1=0 Q2=-5000 Q3=3535.5 Q4=-3535.5 Q5=5000
Q6=0 Q7=0 Q8=2  GOSUB 10000
Q1=Q3 Q2=Q4 Q3=3535.5 Q4=3535.5  GOSUB10000
Q1=Q3 Q2=Q4 Q3=-3535.5 Q4=3535.5  GOSUB10000
Q1=Q3 Q2=Q4 Q3=-3535.5 Q4=-3535.5  GOSUB10000
Q1=Q3 Q2=Q4 Q3=0 Q4=-5000  GOSUB10000
DELAY 500
X0Y0
RETURN
```

PMAC圆弧运动子程序

本子程序自动按圆弧运动.调用此子程序的程序部分只需简单地设置下面一些变量作为运动控制参数.

Q1:圆弧起点的X坐标
Q2:圆弧起点的Y坐标
Q3:圆弧终点的X坐标
Q4:圆弧终点的Y坐标
Q5:圆弧半径
Q6:0为逆时针 1为顺时针
Q7:0为短弧 1为长弧
　　如果规定了圆心则为-1
Q8:角度细分尺寸
如果主机规定了圆心位置,下述变量
也可规定,否则按程序计算
Q9:圆心X坐标
Q10:圆心Y坐标

```
N10000                                  ；圆弧运动子程序
IF (Q7>-1)  GOSUB 20000                 ；必须计算圆心
Q0=Q1-Q9                                ；准备求ATAN2
Q26=ATAN2(Q2-Q10)                       ；从圆心到起点的角度
Q0=Q3-Q9                                ；准备求ATAN2
Q27=ATAN2(Q4-Q10)                       ；从圆心到终点的角度
IF(Q6=0 AND Q27<Q26) Q27=Q27+360        ；正确的方向
IF(Q6=1 AND Q27>Q26) Q27=Q27-360        ；正确的方向
IF(Q6=0)                                ；如果逆时针
  Q28=Q26+Q28                           ；增加角度
  WHILE (Q28<Q27)                       ；不超过终角
    Q11=Q9+Q5*COS(Q28)                  ；圆弧X坐标
    Q12=Q10+Q5*SIN(Q28)                 ；圆弧Y坐标
    X(Q11) Y(Q12)                       ；部分运动
    Q28=Q28+Q8                          ；角度增加量
```

```
  ENDWHILE                                  ; 循环到终角
  X(Q3) Y(Q4)                               ; 终运动
ELSE                                        ; 顺时针圆弧
  Q28=Q26-Q8                                ; 角度减小
  WHILE (Q28>Q27)                           ; 不超过终角
    Q11=Q9+Q5*COS(Q28)                      ; 圆弧X坐标
    Q12=Q10+Q5*SIN(Q28)                     ; 圆弧Y坐标
    X(Q11) Y(Q12)                           ; 部分运动
    Q28=Q28-Q8                              ; 角度减小量
  ENDWHILE                                  ; 循环到终角
  X(Q3) Y(Q4)                               ; 终运动
ENDIF
RETURN

N20000                                      ; 计算圆心的子程序
Q20=SQRT((Q3-Q1)*(Q3-Q1)+(Q4-Q2)*(Q4-Q2))   ; 距离
Q21=Q5*Q5-Q20*Q20/4
IF(Q21<0)                                   ; 运动距离>2R:必须调整FR上
 Q21=0                                      ; 圆心在始点于终点连线
 Q5=Q20/2                                   ; 调整半径为距离的一半
ENDIF
Q23=SQRT(Q21)                               ; FR中点到圆心距离
Q0=Q3-Q1                                    ; 准备求ATAN2
Q24=ATAN2(Q4-Q2)                            ; 始点到终点的角度
Q0=Q20/2                                    ; 准备求ATAN2
Q25=ATAN2(Q23)                              ; 中心离开中线的角度
IF(Q6=1 AND Q7=1 OR Q6=0 AND Q7=0)
  Q26=Q24+Q25                               ; 加上FR到中心的角度
ELSE Q26=Q24-Q25                            ; 减去FR到中心的角度
  Q9=Q1+Q5*COS(Q26)                         ; 中心X坐标
  Q10=Q2+Q5*SIN(Q26)                        ; 中心Y坐标
RETURN
CLOSE
```

例19: 向显示器发送信息

这些PLC程序监测电机位置寄存器,并把它们按比例转换为数值,然后把结果发送到LCD显示.本例描述了一些重要特征.

1. M-变量D用于双字节数值
2. 用外部循环计数器作为计时器
4. 用PMAC内置计时器
5. 24位开放式M-变量包含不同信息用于正确处理图象翻转
6. 在LCD上进行信息显示
7. 在LCD上显示变量值
8. 禁止调用显示命令的条件,以便于下一次扫描过PLC时,显示命令不会自动执行

PLC程序1能用于PLC程序17和18.两程序不同之处在于,程序18用的是PMAC内置计时器,而不是象程序17那样用外部时钟来延时.

```
; ***********************************设置和定义*********************************************************

CLOSE
M0->X:$0,24                                 ; 外部循环计数器(计时)
M85->X:$07F0,24                             ; 打开启动时间存储器
M86->X:$07F1,24                             ; 打开经过时间存储器
M87>X:$0770,1                               ; 电源开/复位时置0位
M90->X:$0700,0,24,S                         ; PMAC时间寄存器
M130->D:$2B                                 ; #1实际位置(4字节)
```

```
M230->D:$67                                   ; #2实际位置(4字节)
M330->D:$A3                                   ; #3实际位置(4字节)
M430->D:$DF                                   ; #4实际位置(4字节)

; ******************************PLC 任务程序*********************************************************************

OPEN PLC 1
CLEAR
DISPLAY 0  " #1 POS=     "                    ; 显示标号
DISPLAY 20  " #2 POS=    "
DISPLAY 40 " #3 POS=     "
DISPLAY 60 " #4 POS=     "
DISABLE PLC 1                                 ; 标号只需写入一次
CLOSE

OPEN PLC 17
CLEAR
IF(M87=0)                                     ; 需要标记起始时间
M85=M0                                        ; 记录起始时间
M87=1                                         ; 不再第一时间通过
ENDIF
P130=M130/(I108*32)                           ; 这些命令用于获得实际位置,加入位置偏差
P230=M230/(I208*32)                           ; 然后按比例转换为数和用户单位
P330=M330/(I308*32)
P430=M430/(I408*32)
M86=M0-M85                                    ; 计算最后一次显示后的时间
IF(M86>P86)                                   ; 超过校正时间?
DISPLAY  9,  10.4, P130                       ; 显示位置
DISPLAY  29, 10.4, P230
DISPLAY  49, 10.4, P330
DISPLAY  69, 10.4, P430
M87=0                                         ; 程序标记起始时间
ENDIF
CLOSE

OPEN PLC 18
CLEAR
IF (M87=0)                                    ; 需要标记起始时间
M90=P86*8388608/I10                           ; 计时存储器赋值P86毫秒
M87=1                                         ; 不再第一时间通过
ENDIF
P130=M130/(I108*32)                           ; 这些命令用于获得实际位置,加入位置偏差
P230=M230/(I208*32)                           ; 然后按比例转换为数和用户单位
P330=M330/(I308*32)
P430=M430/(I408*32)
IF(M90<0)                                     ; 时间到期否?
DISPLAY  9,  10.4, P130                       ; 显示位置
DISPLAY  29, 10.4, P230
DISPLAY  49, 10.4, P330
DISPLAY  69, 10.4, P430                       ; 程序标记起始时间
M87=0
ENDIF
CLOSE
```

装载完程序后,必须设置I5=2或3,然后执行PLC1和17或18.不要同时运行PLC17和18

例20: 在实时位置确定路径

当PMAC执行＂实时位置＂功能时,可能没有用户的确切需要.它可能用于PLC让这些用户构建更精密的功能.这里的路径提供了多种方法来决定系统是否是＂实时位置＂方式.因为每一个用户都有自己构建系统的方式,所以这里提供了很多技术.
在各个应用中,实际的实时位置情况和采用的方法可能不同,这些例子目的就是为如何执行此功能提供一些方法.(这里,输出位以设置)

```
; ***********************************设置和定义*****************************************************************

; 输出位的M-变量定义
M1->Y:$FFC2,8,1                                    ; 设备输出
M2->Y:$FFC2,9,1                                    ; 设备输出
M3->Y:$FFC2,10,1                                   ; 设备输出
M4->Y:$FFC2,11,1                                   ; 设备输出

; 指示没有运动命令的地址位
M140->Y:$0814,0                                    ; #1实时位置
M240->Y:$08D4,0                                    ; #2实时位置
M340->Y:$0994,0                                    ; #3实时位置
M440->Y:$0A54,0                                    ; #4实时位置
M187->Y:$0817,17                                   ; &1实时位置

M77->*                                             ; 自参考变量
M233->X:$0079,13                                   ; #2 要求速度-0 位
M366->X:$00AB,24                                   ; 电机#3实际速度
M466->X:$00E7,24                                   ; 电机#4实际速度

; ***********************执行实时位置功能的PLC程*************************************************************

OPEN PLC 10
CLEAR
```

第一个分支是执行一个简单的电机#1的实时位置功能.当电机#1成为＂实时位置＂时设置一个输出.为了只在实时位置位第一次为真时而不是在电机的整个停留过程设置输出,应用了shadow位(M177).

```
 IF (M140=1)                                       ; 电机在位吗?
 IF(M177=0)                                        ; 最后一次扫描不在?
 M1=1                                              ; 设置输出
 M177=1                                            ; 锁定输出
 ENDIF
 ELSE
 M177=0
 ENDIF
```

第二分支要求在实时位置段进行10次循环,以便于为电机#2报告实时位置

```
IF(M233=1)                                         ; 无运动命令
IF(M240=1)                                         ; 计数器增值
P240=P240+1                                        ; 10次?
IF(P240!<10)                                       ; 防止溢出
P240=10
M2=1                                               ; 设置实时位置信号
ELSE
M2=0                                               ; 清除实时位置信号
ENDIF
ELSE                                               ; 不在范围内
P240=0                                             ; 计数器复位
M2=0                                               ; 清除实时位置信号
ENDIF
ELSE                                               ; 命令运动
P240=0                                             ; 计数器复位
```

```
M2=0                                                    ; 清除实时位置信号
ENDIF

第三分支要求电机#3和#4都在位,而且自上一循环设置输出后运动少于范围的1/4
IF(M187=1)                                              ; 所以C.S.1电机在位?
P366=M366/(I309*32)                                     ; #3速度
P466=M466/(I409*32)                                     ; #4速度
IF(P366<(I328/64 AND P466<I428/64))                     ; Ix28 in16ct
M3=1
ENDIF
ELSE
M3=0
ENDIF
CLOSE

I5=3                                                    ; PLC能够运行

ENABLE PLC 10
```

例21:编码器模数检测

本PLC程序应用了PMAC的位置捕捉功能,用第3个通道触发捕捉装置,然后检测自最后一次触发的计数值,就可以对编码器1进行一圈检测.这个动作只有在通道C被捕捉后才能开始.

```
; ******************************定义和设置*********************************************************************

CLOSE                                                   ; 确保在线
P50=2000                                                ; 每转时的计数值
I902=1                                                  ; 在通道C上升沿捕捉
M40->*                                                  ; 控制操作标志
M41->*                                                  ; 错误标志
M103->X:$C003,0,24,S                                    ; 编码器1捕捉位置寄存器
M117->X:$C000,17,1                                      ; 编码器1位置捕捉标志
CLOSE                                                   ; 确保其他关闭

; ******************************PLC程序************************************************************************

OPEN PLC 4                                              ; 准备编辑
CLEAR                                                   ; 清除存在的内容
IF (M40>0)                                              ; 通过设置M40为1来启动
IF(M40>1 AND M117=1)                                    ; 非第一次,捕捉
P12=P11                                                 ; 从最后一次存储
P11=M103                                                ; 最新捕捉
P13=P11-P12
; 两次捕捉之间的差距IF(P13>P50/2)                                                ; 正转
        IF(ABS(P13-P50)>1)                              ; 多于1OFF?
                M41=1                                   ; 设置出错标志
        ENDIF
    ELSE
        IF(P13<-P50/2)                                  ; 正转
            IF(ABS(P13+P50)>1)                          ; 多于1OFF?
                M41=1                                   ; 设置出错标志
            ENDIF
        ELSE                                            ; 同一指示第二次读入
            IF(ABS(P13)>6)                              ; 出错范围容许脉冲宽度
                M41=1                                   ; 设置出错标志
            ENDIF
        ENDIF
```

```
    ENDIF
  ELSE                                    ; 第一次THRU
   IF(M117=1)                             ; 位置捕捉否?
    P11=M103                              ; GRAB捕捉位置
    M40=2                                 ; 标记第一次完成
   ENDIF
  ENDIF
ENDIF
CLOSE
M40=0                                     ; 不会立即开始
M41=0                                     ; 清除出错标志启动
ENABLE PLC 4
```

例22: 存储起始位置编码器偏移量

本程序存储起始位置编码器偏移量以备以后使用。当找到起始位置时,在开启/复位状态下编码器位置为0,电机位置设置到0,但编码器位置不是。因此,当我们在位置比较和捕捉功能里直接用到编码器寄存器时,就需要知道电机和编码器位置偏移量(在很多情况下控制系统自动处理偏移量)。一旦发现位置,PLC程序用P-变量来存储偏移量(位置捕捉寄存器将再次用到)。

```
; ****************************设置和定义*********************************************************************

CLOSE
M103->X:$C003,0,24,S                      ; 编码器1位置捕捉寄存器
M140->X:$003的0,1                         ; #1逐步位置查找
M141->*

; ***************************PLC程序清单*********************************************************************

OPEN PLC 3
CLEAR
IF(M140=1)                                ; 寻找起始坐标
  M141=1                                  ; 设置标志
ENDIF
IF(M140=0 AND M141=1)                     ; 当前发现起始标志
  P103=M103                               ; 保存捕捉起始偏移量
  M141=0                                  ; 标记没有正在寻找
ENDIF
CLOSE
ENABLE PLC 3
```

例23: 纯位置运动

本程序通过电机#1进行纯位置运动。因为要插入运动程序,所以它通过PLC直接写入给定位置寄存器来实现。计数形式的运动距离必须放在变量P80中,毫秒级的运动时间必须放在P85中。要在不同的电机上运行此程序,只需改变M81即可。

```
CLOSE                                     ; 确保在线
DELETE GATHER                             ; 确保THIS IS ROOM
M88->X3,18                                ; 数据采集预备
M89->X3,19                                ; 数据采集中
M0->X0,24                                 ; SERVO 循环计数器(计时)
M85->X$07F0,24                            ; 打开存储器计时
M86->X$07F1,24                            ; 打开ELAPSED存储器
M81->D$28                                 ; #1CMD.位置(双字)

OPEN PLC 2
CLEAR
CMD " GAT "                               ; 打开数据采集
```

```
P81=P80*I108*32                          ; 计算位置寄存器增量
P86=P85*8388608/I10                      ; SERVO循环中的步进时间
M81=M81+P81                              ; 给定位置增量
M85=M0                                   ; 计时器启动
M86=M0-M85                               ; 时间差
WHILE(M86<P86)                           ; 仍在等规定时间?
M86=M0-M85                               ; 时间差
ENDWHILE
M81=M81-P81                              ; 给定位置减少量
M85=0                                    ; 重启计时器
M86=M0-M85                               ; 时间差
WHILE(M86<P86)                           ; 超过规定时间?
M86=M0-M85                               ; 时间差
ENDWHILE
CMD " ENDGATHER "                        ; 关闭数据采集
DISABLE PLC 2                            ; 关闭 不重复
CLOSE
I5=3                                     ; 确保PLC能动作
```

例24: 电机脉冲输出

这些简单的程序结合起来,在编码器1到达设置位置时产生一系列的脉冲。它用DSP GATE IC中的位置比较特征来产生脉冲。PLC0用来UPDATE预装比较位置(如果需要非常快的脉冲--接近1KHZ,就需要快速PLC0)。PLC1用于准备功能,只执行一次就关闭。

```
; ********************************设置和定义*************************************************************

CLOSE
M1O1->X:$C001,0,24,S                     ; 编码器1实际位置
M1O3->X:$C003,0,24,S                     ; 编码器1比较位置
M1O5->X:$C7F0,0,24,S                     ; SCRATCH 24位寄存器
                                         ; (处理ROLLOVER)
M111->X:$C000,0,11,1                     ; ENC1 EQU标志LATCH控制
M112->X:$C000,0,12,1                     ; ENC1 EQU输出使能
M116->X:$C000,0,16,1                     ; ENC1 EQU标志位
P101=50                                  ; 计数增值

; ********************************PLC程序*****************************************************************

OPEN PLC 1
CLEAR
M112=1                                   ; EQU输出使能
M105=M101+P101                           ; 第一个EQU位置送入SCRATCH寄存器
                                         ; (可以是你需要的任何位置)
                                         ; 这里和起始位置有关
M103=M105                                ; 比较寄存器赋值
M105=M105+P101                           ; 准备下一个值
ENABLE PLC 0                             ; 启动 REPETITIVE FN
DISABLE PLC 1                            ; 仅一次
CLOSE

OPEN PLC 0
CLEAR
IF(M116=1)                               ; 到达最后的EQU位置?
  M103=M105                              ; 装载最后的EQU位置
  M105=M105+P101                         ; 准备下一个EQU位置
  M111=0                                 ; 清除和设置控制位
  M111=1                                 ; 清除LATCHED标志
```

```
ENDIF
CLOSE
```

例25: PLC I/O

本例说明怎样应用PMAC一般I/O来运行PLC程序。

```
; *********************************定义和设置*************************************************************

CLOSE                                                    ; 确保在线
M1->Y:$FFC2,8,1                                          ; 设备输出1
M2->Y:$FFC2,9,1                                          ; 设备输出2
M3->Y:$FFC2,10,1                                         ; 设备输出3
M4->Y:$FFC2,11,1                                         ; 设备输出4
M11->Y:$FFC2,0,1                                         ; 设备输入1
M12->Y:$FFC2,1,1                                         ; 设备输入2
M13->Y:$FFC2,2,1                                         ; 设备输入3
M14->Y:$FFC2,3,1                                         ; 设备输入4
M20..39->*                                               ; 自参照标志变量

; *********************************PLC程序*****************************************************************

CLOSE                                                    ; 确保其它缓存关闭
OPEN PLC 5                                               ; 打开编辑缓存
CLEAR                                                    ; 清除存在的内容

第一分支按设备输入1设置变量.变量可以是终点,速度,时间等

IF(M11=1)                                                ; 设备输出1正确否?
  P1000=5000                                             ; 若正确,设置此值
ELSE
  P1000=500                                              ; 若不正确,设置此值
ENDIF

下一分支在每次设备输入2正确时使计时器增值(边缘触发)

IF(M12=1)                                                ; 设备输出2正确否?
 IF(M22=0)                                               ; 最后一个时区不正确
   P8=P8+1                                               ; 上升沿到来,增值
   M22=1                                                 ; 注释下一时区正确
 ENDIF
ELSE                                                     ; MI2不正确
 M22=0                                                   ; 注释下一时区不正确
ENDIF

下一分支在所有的4个标志变量为真时设置输出

IF (M23 AND M24=1)                                       ; 前两个标志为真?
AND (M25 AND M26=1)                                      ; 后两个标志为真?
  M1=1                                                   ; 设置输出
ELSE
 M1=0                                                    ; 否则清除输出
ENDIF
CLOSE                                                    ; 关闭缓存
ENABLE PLC 5                                             ; 运行程序
```

例26: 圆弧插补

本程序在XY平台上“画”一个菊花瓣形状的图。它举例说明了圆弧插补,应用子程序和输出设置

```
OPEN PROG 10 CLEAR
HM1..8                                  ; C.S所有电机复位
F2000                                   ; 设置回馈率
TA 10                                   ; 加速时间
NORMAL K-1                              ; 定义XY平台为园
LIN                                     ; 线性运动启动
GOTO 0                                  ; 跳过子程序
N62                                     ; 子程序开始标号
DWELL01
M1=0                                    ; 关闭输出
F2000                                   ; 快速回馈率
DWELL 100
RET                                     ; GOSUB后返回
N63                                     ; 启动子程序标号
DWELL01
M1=1                                    ; 打开输出
F1000                                   ; 低速回馈率
DWELL 100
RET                                     ; GOSUB后返回
N0                                      ; 主程序启动
GOSUB 62
X2082.4000 Y1778.0000                   ; 线性偏移运动
GOSUB 63                                ; 打开输出

; 整圆运动(终点=起点)

CIR 1 X2082.4000 Y1778.0000 I-606.2000 J0.0000 LIN
GOSUB 62                                ; 关闭输出
X1880.9716 Y2232.2616
GOSUB 63

; 第一 " 瓣 "
CIR 2 X1070.7003 Y2233.1163 I-404.7716 J345.5384 LIN
GOSUB 62
X1930.4225 Y1383.0427
GOSUB 63

; 第二 " 瓣 "
CIR 2 X1937.9537 Y2179.42222 I345.5775 J394.9573 LIN
GOSUB 62
X1012.0638 Y2163.0278
GOSUB 63

; 第三 " 瓣 "
CIR 2 X1013.4259 Y1394.1638 I-335.6638 J-385.0278 LIN
GOSUB 62
X1071.4245 Y1323.7339
GOSUB 63

; 第四 " 瓣 "
CIR 2 X1885.0458 Y1318.9081 I404.7755 J-345.5339 LIN
GOSUB 62
X74.0000 Y178.2000
GOSUB 63
X2870.6000 Y178.2000
X2870.6000 Y3391.2000
X74.0000 Y3391.2000
X74.0000 Y178.2000
```

```
GOSUB 62
X0.0000 Y0.0000
CLOSE
```

例27: 成比例电压输出

本例应用额外的模拟输出来提供轴位置的电压比例
注意；确保没有正在运行的电机用这些DACs

```
; ************************************定义和设置*****************************************************************

M101->X:$C001,0,24,S                                    ; 编码器1位置
M201->X:$C005,0,24,S                                    ; 编码器2位置
M301->X:$C00B,8,16,S                                    ; DAC3输出
M401->X:$C00A,8,16,S                                    ; DAC4输出
M30->*                                                  ; 自参考标志
; P10=?                                                 ; (比例常量)

; ***********************************PLC程序*************************************************************************

CLOSE                                                   ; 关闭所有已打开的缓存
OPEN PLC 3                                              ; 打开缓存编辑
CLEAR                                                   ; 清除缓存内容
IF(M30=1)                                               ; M30设置模式
  M302=P10*M101                                         ; DAC3和位置1成比例
  M402=P10*M201                                         ; DAC4和位置2成比例
ELSE                                                    ; 非此模式
  M302=0                                                ; DAC3输出0
  M402=0                                                ; DAC4输出0
ENDIF
CLOSE                                                   ; 关闭缓存
ENABLE PLC 3                                            ; 启动此PLC程序
```

例28:设备输出的脉冲输出

本例的PLC程序,在带编码器1和2的轴每次走过250个计数距离时,就用脉冲进行设备输出。下一次通过PLC时脉冲关闭(~1msec)。

```
; ************************************设置和定义*****************************************************************

M1->Y:$FFC2,8,1                                         ; 设备输出1
M20->*                                                  ; 自参考标志变量
M101->X:C001,0,24,S                                     ; 24位编码器1位置
M201->X:C005,0,24,S                                     ; 24位编码器2位置

; ***********************************PLC程序内容*********************************************************************

CLOSE                                                   ; 关闭所有打开的缓存
OPEN PLC 2                                              ; 打开缓存编辑
CLEAR                                                   ; 清除旧内容
IF(M20=1)                                               ; M20设置模式
  IF(M1=0)                                              ; 输出关闭
    P102=(M101-P101)*(M101-P101)                        ; X距离平方
    P202=(M201-P201)*(M201-P201)                        ; Y距离平方
    IF(SQRT(P102+P202)>250)                             ; 距离>250?
      M1=1                                              ; 设置输出
      P101=M101                                         ; 复位最后位置变量
      P201=M201                                         ; 同上
```

```
    ENDIF
  ELSE
    M1=0                                    ; 下一次清除输出
  ENDIF
ENDIF
CLOSE                                       ; 关闭缓存
ENABLE PLC 2                                ; 运行
```

例29:电机频繁启动的PLC程序

本程序显示怎样用微动开关输入来产生特殊电机专用的频繁启动开关。注意特殊用到 " latching标志 " 以便于只有在输入改变时才发出命令。使它们边缘触发。

```
; ********************************设置和定义*************************************************************

CLOSE
M50->Y:$FFC1,0                              ; 微动开关输入位0
M51->Y:$FFC1,1                              ; 微动开关输入位1
M52->Y:$FFC1,2                              ; 微动开关输入位2
M53->Y:$FFC1,3                              ; 微动开关输入位3
M60->*                                      ; M50 Latching位
M61->*                                      ; M51 锁定位
M62->*                                      ; M52 锁定位
M63->*                                      ; M53 锁定位

; ***********************************PLC程序**********************************

OPEN PLC 16
CLEAR
IF(M50)=1                                   ; 电机1 jog plus开启
  IF(M60=0)                                 ; 但不是最后一次开启
    COMMAND " #1J+ "                        ; 发出命令
    M60=1                                   ; 设置锁定位
  ENDIF
ELSE                                        ; 电机1 jog plus关闭
  IF(M60=1)                                 ; 但不是最后一次关闭
    COMMAND " #1J/ "                        ; 发出停止命令
    M60=0                                   ; 设置锁定位
  ENDIF
ENDIF

IF(M51)=1                                   ; 电机1 jog plus开启
  IF(M61=0)                                 ; 但不是最后一次开启
    COMMAND " #1J+ "                        ; 发出命令
    M61=1                                   ; 设置锁定位
  ENDIF
ELSE                                        ; 电机1 jog plus关闭
  IF(M61=1)                                 ; 但不是最后一次关闭
    COMMAND " #1J/ "                        ; 发出停止命令
    M61=0                                   ; 设置锁定位
  ENDIF
ENDIF

IF(M52)=1                                   ; 电机2 jog plus开启
  IF(M62=0)                                 ; 但不是最后一次开启
    COMMAND " #2J+ "                        ; 发出命令
    M62=1                                   ; 设置锁定位
  ENDIF
ELSE                                        ; 电机2 jog plus关闭
```

```
  IF(M62=1)                                    ; 但不是最后一次关闭
    COMMAND " #2J/ "                           ; 发出停止命令
    M62=0                                      ; 设置锁定位
  ENDIF
ENDIF

IF(M53)=1                                      ; 电机2 jog plus开启
  IF(M63=0)                                    ; 但不是最后一次开启
    COMMAND " #2J+ "                           ; 发出命令
    M63=1                                      ; 设置锁定位
  ENDIF
ELSE                                           ; 电机2 jog plus关闭
  IF(M63=1)                                    ; 但不是最后一次关闭
    COMMAND " #2J/ "                           ; 发出停止命令
    M63=0                                      ; 设置锁定位
  ENDIF
ENDIF
```

例30:开环运动

本例显示如何运用PMAC运动程序获得有效的开环动作。在开环方式下,可以直接对DAC发出命令。当用到电流环放大器时,输出是力或力矩命令。当用到速度环放大器时,输出是速度命令。在此开环运动中,PMAC技术上仍在试图闭合位置环路。然而,通过设置比例为0,PID输出就总为0。这意味着写到输出偏移量I-变量(Ix29)的值直接输出到DAC上。这只对没经过PMAC整流的电机不适用。

必须仔细处理的是这种方式的输入输出转换。进入开环,DAC输出就拷贝到偏移量变量,比例系数设置为0(对最平滑转换而言)。另外,微分系数(Ix33)和微分限(Ix63)也设置为0,所以它们就不起作用,而且下面的误差限也设置为0来使其失效。回到闭环模式,实际的电机位置必须拷贝到两个给定位置寄存器中,一个用于电机,另一个用于轴,所有再闭环时就不会摆动。

本例中,10步后再次闭环,减少偏移量增加比例系数。在一些特殊应用中可能不需要。但是,可能的最平滑转换是倾斜的目的。如果需要的话,微分系数和出错极限也重新存储。此例正好在协调系统1中用电机1驱动X轴。这项技术可以扩展到所有系统的所有电机中。

在有些情况下,通过CMD程序命令直接向电机发出开环指令可以避免应用此程序。如果开环电机被分配到一个没有运行运动程序的坐标系统中时,这是可能的。

```
; *********************************设置和定义*************************************************************

CLOSE
M102->Y:$C003,8,16.S                           ; DAC1输出寄存器
M162->D:$002C                                  ; 电机1实际位置寄存器
M163->D:$080B                                  ; 电机1目标位置寄存器
M165->L:$081F                                  ; C.S.1X轴标定目标位置

; ****************************运动程序举例*************************************************************

PROG 12
CLEAR
WHILE(1<2)                                     ; 无限循环
```

程序的第一部分是一个显示闭环运动的“虚拟”路径。实现一个常值运动。使其保持这种方式的条件是人为的,肯定与实际应用完全不同。

```
    WHILE (M1=0)                               ; 保持循环直到条件为假
      INC                                      ; 增值运动
      TM50
      X100
    ENDWHILE
```

```
DWELL50                    ；确保停止
DWELL50                    ；这样变量值不会改变得太快
I129＝M102                 ；输出偏移量＝数模转换器输出
I130＝0                    ；比例增益设为零
I163＝0                    ；累计极限
I133＝0                    ；累计增益
I111＝0                    ；关闭下面的错误极限
```

这部分程序构成“开环”方式。注意PMAC认为当前处于停止状态。在此期间主机或可编程中断控制器可以写变量I129，改变开环输出。

```
WHILE (M1=1)               ；保持循环直至失败
   M163＝M162              ；使目标位置＝实际位置
   DWELL10                 ；保持一定时间；其它轴可运动
ENDWHILE
```

这部分程序转换回真正的闭环。可以放在子程序中。

```
M163 ＝M162                ；电机目标位置＝实际位置
M165 ＝M162/(I108*32)      ；轴标度目标位置＝实际位置
DWELL5                     ；确保上面的语句执行
DWELL5                     ；在增益被重新说明之前
P129 ＝I129                ；保持当前输出值
P130 ＝8000                ；支增益增量（总量的1/10）
P131＝ 0                   ；增量计数
I163＝ 8000000             ；恢复累计极限值
I133＝ 50000               ；恢复累计增益
WHILE (P131<10)            ；循环十次
    I129＝ I129 －P129/10  ；减去初值的1/10
    I130＝ I130+P130       ；增加目标值的1/10
    P131＝ P131+1          ；累计循环次数
    DWELL20                ；控制转换速度
ENDWHILE
I129＝ 0                   ；确保获得正确的终值
I111 ＝32000               ；恢复下面的错误极限（2000 脉冲）
```

程序的最后部分停止于真正的闭环，这样可以判断出过渡过程。同上，这段程序的条件是随机的。

```
WHILE  (M2＝ 0)
   DWELL50
ENDWHILE
M2＝ 0
ENDWHILE
CLOSE
```

例31:　反相运动学

这个例子表明了如何使用PMAC板执行反相运动--
用户坐标系（通常为直角坐标系）到电机坐标系或复合坐标系的转换。这个程序只工作于两轴系统—它演示了如何将XY坐标转换为径坐标和角坐标。这样可以在转台上以旋臂定义点。概念可以扩展到六轴甚至八轴的系统—
计算更加复杂。注意逻辑分支，不是纯粹的换算，可以在转换中完成（这里是处理C轴倾向倒转）。

```
；**********初始化与定义**********************************************************************************

#3 －>5000U                ；径坐标定义；单位应该
                           ；同XY坐标单位相同
#4 －>13.889C              ；角坐标（theta）定义；单位
```

```
                                    ；可以是度或弧度，但必须与I15匹配。
                                    ；这里5000个脉冲/转 等同于13.889个脉冲/度
I15=0                               ；在三角计算中使用度
Q21=5                               ；X 零点在U轴坐标系中的位置
Q22=0                               ；C轴偏移量
```

反相运动的程序实际上是在复合坐标系执行。这成为子程序并预期设定‘X’坐标（在Q124中）和’Y’坐标（在Q125中）。它可以将这个坐标转换为极坐标系（在反相运动问题中）的坐标并发出运动指令—这里为U轴(半径)和C轴（角度）。Q21和Q22应分别提供U轴和
C轴的偏移量。运动被分为若干小段，在每段分支中进行转换。得出的相邻段轨迹被接合在一起以产生光滑而精确的轨迹。

```
CLOSE
OPEN PROG 500
CLEAR
READ(X,Y)                           ；X终点—> Q12 4；Y终点—> Q125
Q12=Q10*Q11/1000                    ；区段距离（用户单位）
Q14=Q24                             ；X工作点是X出发点
Q15=Q25                             ；Y工作点是Y出发点
SPLINE1                             ；三次样条模式
TA(Q11)                             ；区段时间
Q16=0                               ；开始循环
WHILE (Q16<1)
```

这段 程序将线性运动分为若干小段，坐标转换在每段分支中进行。下一个计算只需针对“进给轴”。

```
Q34=Q124-Q14                        ；X方向要走的距离
Q35=Q125-Q15                        ；Y方向要走的距离
Q36=SQRT(Q34*Q34+Q35*Q35)           ；要走的向量距离
IF (Q36>Q32)                        ；比区段距离大吗？
    Q37=Q12/Q36                     ；要走的部分距离
```

下面的插值应计算所有的轴，无论是否为进给轴

```
Q14=Q14+Q34*Q37                     ；下一段X轴终点位置
Q15=Q15+Q35*Q37                     ；下一段Y轴终点位置
ELSE                                ；最后一个运动区段
Q14=Q124                            ；下一区段结束运动
Q15=Q125                            ；下一区段结束运动
Q24=Q124                            ；下个运动的X初始点
Q25=Q125                            ；下个运动的初始点
Q16=1                               ；标记最后一段
ENDIF
```

现在转换到复合坐标系

```
Q17=Q21-SQRT(Q14*Q14+Q15*Q15)       ；径向坐标
Q28=Q18                             ；上一个角坐标
Q0=Q14                              ；ATAN2的第二个参数
Q18=ATAN2(Q15)+Q22                  ；角坐标
IF  (Q18-Q28>180)   Q8=Q8-360       ；滚改器换为反向
IF  (Q18-Q28<-180)  Q8=Q8+360       ；滚改器换为正向
U(Q17) C (Q18+Q8)                   ；执行该段运动
ENDWHILE
CLOSE
```

下面是使用上面程序的主程序的例子。该程序实现简单的矩形运动。它基本上看不到反相运动学问题。这表明实际的反相运动学的问题被最终程序员大量隐藏起来。

```
OPEN PROG 25
CLEAR
Q10=2                                  ; 期望的进给速度（XY 单位/秒）
Q11=30                                 ; 插补的间歇时间
CALL 500 X2 Y2                         ; 运动到矩形的初始点
DWELL100
M1=1                                   ; 打开输出（例如：激光）
CALL 500 X2 Y-2                        ; 做矩形的第一条边
DWELL100
CALL 500 X-2 Y-2                       ; 做矩形的第二条边
DWELL100
CALL 500 X-2 Y2                        ; 做矩形的第三条边
DWELL 100
CALL 500 X2 Y2                         ; 做矩形的最后一条边
DWELL100
M1=0                                   ; 关掉输出
CALL 500 X-5 Y0                        ; 返回退刀位置
CLOSE
```

启动数据；仅第一次运行前使用

```
Q24=-Q21                               ; 起动X-值
Q25=-Q22                               ; 起动Y-值
Q8=0                                   ; 起动C轴倾向倒转
Q18=0                                  ; 起动C轴
```

例32：旋转头的定位

这个例子表明如何在XY
系统的角运动中定位旋转头。此处关键的概念在于使用ATAN2函数，函数返回一个介于-180度与+180度之间的值。ATAN2函数使用两个参数：第一个是Y-轴值，第二个是X-轴值，值放在坐标系统中的变量Q0中。

这种方法将切割头放在与XY轴坐标系不同的一个PMAC坐标系中。因此可以运行与XY系统分离的运动程序。无论XY程序如何，头程序都可以相同—
它仅使用XY系统中的实时数据来规划自己的运动。这也使得XY程序可以自动生成而无需考虑头运动的细节。随着头程序驻留在PMAC中，头运动根据XY程序中的线索自动执行。

这个例子头程序查找XY程序中的两个专门的线索：DWELLs和M1状态。在XY
DWELL期间，将拉升旋转头（如果它还未被拉升）。头程序还将发送一个'%0'命令到XY
系统，这样下一个XY运动将不会在头程序准备好前就开始。当XY DWELL执行结束后（DWELL不受%
命令的影响），头程序将查看是否设置好了M1（XY程序中表明刀具向下的标志）。如果已经设置，则头程序使用已计算好的XY目标位置计算所需的旋转头取向，移动旋转头到该角度，降下刀具头，再重新开始XY 程序（使用一条 %100 命令）。

```
; ************************设置与定义*****************************************************************************

CLOSE
I13=10                                 ; 运动区段时间（毫秒）
I15=0                                  ; 三角函数使用单位为度
I427=14400
                                       ; 4号电机速度14400个脉冲/转，因此每14400个脉冲后倾向倒转一次
I194=2000000                           ; 坐标系一次基本转换速率
M161→D: $0028                          ; 一号电机当前的给定位置
M163→D: $080B                          ; 一号电机目标位置
M261→D: $080B                          ; 二号电机当前的给定位置
M263→D: $08CB                          ; 二号电机目标位置
```

```
M150→X: $0818,0                          ; 坐标系1程序运行状态字节
M151→X: $0038,15                         ; 一号电机停止过程字节
M1→Y: $FFC2,8                            ; 机器输出1（PMAC A到D通道开/关信号）

&1                                       ; XY坐标系
#1→10000X                                ; 一号电机在坐标系1中为X轴
#2→10000Y                                ; 二号电机在坐标系1中为Y轴
&2
#3→1000Z                                 ; 三号电机在坐标系2中为Z轴（垂直轴）
#4→40C                                   ; 四号电机在坐标系中为旋转轴（单位为度）
OPEN PROG 2
CLEAR
FRAX (Z,C)                               ; 令垂直轴和旋转轴做进给运动
ABS                                      ; 滚改器以绝对模式恰当工作
Q10=I108*32                              ; 编码器1以Mx 61，Mx 63单位计数
CMD “&1%0”                               ; 暂停坐标系1中的轴运动
F10
C0                                       ; 令旋转轴回到开始位置
DWELL5
Q2=0                                     ; 开始角度计算值
CMD“&1%100”                              ; 存贮坐标系1的运动
WHILE (M150=1)                           ; 只要XY程序在执行中
    IF (M151=1)                          ; 在进程中XY 停止
        CMD “&1%0”                       ; 在开始时保存下一个XY运动
        F5
        Z0                               ; 拉起旋转头
        WHILE (M151=1) WAIT              ; 等待XY 停止命令执行结束
        IF (M1=1)                        ; XY程序询问旋转头是否降下
            GOSUB 1000                   ; 计算头的角度
            F10
            C (Q2)                       ; 旋转头到开始角度
            Z10                          ; 降下旋转头
        ENDIF
        DWELL5
        DWELL5                           ; 这样旋转头的上下运动结束
        CMD“&1%100”                      ; 允许XY运动继续
    ENDIF
    TM20                                 ; 设置20毫秒运动
    WHILE (M150=1 AND M151=0)            ; XY程序运行；没有停顿
        GOSUB 1000                       ; 计算旋转头的角度
        C (Q2)                           ; 将旋转头移动到新的角度
    ENDWHILE
ENDWHILE
RETURN                                   ; 当XY程序结束时头程序随之结束
N1000                                    ; 开始头角度计算程序
Q0=M163-M161                             ; X目标位置-X当前位置
Q1=M263-M261                             ; Y目标位置-Y当前位置
IF (ABS (Q0)>Q10 OR ABS(Q1)>Q10)         ; 目标位置不=当前位置?
    Q2=ATAN2 (Q1)                        ; 计算角度—ATAN2（Q1，Q0）
ENDIF                                    ; 注意目标位置是否=当前位置
                                         ; Q2不变

RETURN
```

例33：时间基准跟随

这是一个打盘机的例子。它演示了如何使用PAMC的时间基准跟随模式将一根轴做为另一根轴（或几根其它轴）的主控轴。在这个应用实例中，一号电机（坐标系1）是型心轴，并做为主控轴。坐标系2

从属于一号电机编码器的频率（这个编码器用作一号电机的反馈元件和决定坐标系2的主控频率）。这里，只有坐标系2中的一根轴：横向轴（电机3），沿着型心轴在轴线方向来回运动放置金属线。

正确缠绕 （正确的螺旋线或类似的应用）
的关键在于保持控制良好的侧伏角，这需要使切割轴和型心轴有一个合理的速比。编写一个简单的协调程序可以实现，但是通过这样一个程序在型心轴加速或减速时使轴保持正确的速比是困难的。实践基准跟随机制使得程序更加简单：可以假设型心轴总以固定（实时）频率旋转对切割轴编程，但是程序执行时以一个与实际频率有某个比例关系的速率。第一件要做的事情是建立比例因子来设置轴之间转速正确的比率。

```
; ************************示范系统参数*********************************************************************
; 型心轴：
; 分辨率：100线道/转=400计数脉冲/转
; 最大速度: 15000转/分=100计数脉冲/毫秒
; 最小转换速度:100转/分=0.667计数脉冲/毫秒
; “实时”频率=64计数脉冲/毫秒 ( 希望幂为2；希望最大不超过200%)
; 比例因子:131,072/64=2408
; 在“实时”：6.25毫秒/转

; 切割轴:
;  分辨率:1000线道/英寸=4000计数脉冲/英寸

; ************************PMAC设置与定义*****************************************************************

CLOSE
I103=$720                       ; 一号电机使用编码器1做反馈(=1824D)
WY$728,$400720                  ; 从编码器1创建时间基准信息
WY$729,2048                     ; 时间基准的比例因子
&1
#1->400X                        ; 一号电机是坐标系中的X轴, X单位是转数  I190=60000
                                ; 坐标系1进给速率时间单位为分钟
I193=$0806                      ; 坐标系1使用固定时间基准(=2054D)

I303=$722                       ; 三号电机使用编码器3反馈(=1825D)
&2
#3->4000Y                       ; 电机3代表坐标系2中的Y轴；Y轴单位为英寸
I293=$08C6                      ; 坐标系2使用固定时间基准(=2246D)
                                ; 这是起始—如复位时的设置
                                ; 当程序开始后,该变量变为$0729
                                ; (=1833D)—来自编码器1的变量时间基准
I294=8388607                    ; 用于立即改变坐标系2 的%值
Q3=6.25                         ; “实时”中的毫秒/转

CLOSE

; ************************型心轴程序*********************************************************************

OPEN PROG 1                     ; 型心轴程序: 一个长时间的运动
CLEAR
I293=$729                       ; 使坐标系2从属于编码器1的频率
TS0                             ; 无S形曲线加速过程
TA(Q5*1000)                     ; 用坐标系1的参数Q5设置加速时间(sec)
F(Q6)                           ; 用坐标系1的参数Q6设置进给速度(rpm)
INC                             ; 增量模式 ,因此不必考虑初始点
X(Q7)                           ; 用坐标系1的Q7设置转数的#
DWELL10                         ; 允许停止
DWELL10                         ; 在改变坐标系2的从属地位之前
I293=$08C6                      ; 将坐标系2返回到内部的时间基准
CMD“&2Q”                        ; 忽略切割程序
```

```
DWELL500                        ; 使之稳定
M1=1                            ; 打开输出以切割金属线
CMD “#3HM”                      ; 令切割电机复位
CLOSE

; **********************切割轴程序*********************************************************************

CLOSE
OPEN PROG 2
CLEAR
I293=$729
TS0
TA3                             ; 短的加速/减速时间
Q1=0                            ; 从零开始计数
Q2=Q519                         ; 坐标系2中Q519类似于坐标系2中Q7(#  of turns)
TM(Q3*Q12)                      ; 时间/层=(时间/转数)*(转数/层)
WHILE (Q1<Q2)
    Y(Q11*Q12)                  ;  距离等于螺距乘以转数
    DELAY0                      ; 完成全部行程
    Q1=Q1+Q12                   ; 改变计数值
    IF (Q1<Q2)
        Y0                      ; 复位
        DELAY0                  ; 完成全部行程
        Q1=Q1+Q12               ;  行进了两转
    ENDIF
ENDWHILE
CLOSE

; ******************为一个专门的卷绕应用设定参数*****************************************************

CLOSE
&1                              ; 型心轴参数
Q5=5                            ; 5秒加速/减速时间
Q6=2000                         ; 2000rpm
Q7=3000                         ; 总共3000转

&2                              ; 旋转参数
Q11=0.005                       ; 螺距5/1000英寸
Q12=200                         ; 200转数/层 (1英寸宽)
```

例34: 运动到触发开关

这个例子演示了如何进行定位运动,
通常即为所了解的在PMAC上“运动到触发开关”。它使用一个位置寄存器来立即锁存触发位置。程序可以在不损失任何精确性的情况下稍后响应触发。在响应触发时，PMAC必须获取被捕捉的编码器位置,在加上初始偏移量(见HOMOFFST。PMC)后转换为电机位置,然后将其分划为轴的位置,从我们希望停下的触发位置加入期望距离。

```
; *************************设置与定义*********************************************************************

M103->X:$C003,0,24,S            ; 编码器1捕捉位置寄存器
M117->X:$C000,17,1              ; 编码器1位置-捕捉标志
I902=2                          ; 编码器1捕捉在上升标志沿
I903=0                          ; 编码器1使用HMFL1进行捕捉
I190=1000                       ; 进给速度  units/sec
M191->L:$0822                   ; 一号电机X轴系数(S.F.)
M194->L:$0825                   ; 一号电机轴偏移量

; *************************电机程序*********************************************************************
```

```
OPEN PROG 3
CLEAR
ABS                                  ; 绝对运动定义
TA50                                 ; 加速时间
TS0                                  ; 无S形曲线加速
F10                                  ; 10用户 units/sec
P100=5                               ; 未及触发位置
X(P100)                              ; 快速运动部分—加速

; 现在开始循环等待触发
TM10                                 ; 循环时间是10毫秒
TA10                                 ; 加速时间必须<=TM
P99=M103                             ; 确保它已被清除
WHILE  (M117=0)                      ; 循环直至捕捉位置
    P100=P100+0.1                    ; 增量给定位置
    X(P100)                          ; 保持运动
ENDWHILE

; 发现触发器—计算公式定义
P104=M103-P103                       ; 电机位置=编码器位置-原始偏移量
P105=(P104-M194)/M191                ; 轴位置=(电机位置-轴偏移量)/S.F.

; 执行最后定位运动
TA50
F10                                  ; 以10 单位每秒匹配旧速度
X(P105+0.75)                         ; 在距触发器3/4单位时停止
DWELL100                             ; 保持在那儿

; 在此可以执行更多的动作
CLOSE
```

例35: 线性和圆弧运动

这个运动程序是一个在XY平台做直线和圆弧运动的简单程序。它画出一个带圆角的矩形。

```
CLOSE
I13=0                                ; 圆弧的运动分段时间
OPEN PROG 55
CLEAR
NORMAL K-1                           ; XY平台上的圆
RAPID                                ; 快速运动模式
X10Y5                                ; 运动到起始位置
M1=1                                 ; 打开输出(如:激光, 喷水)
LINEAR                               ; 线性内插模式
F10                                  ; 定义速度
X30Y5                                ; 矩形底边

CIRCLE2                              ; 反时针画圆
X35Y10J5                             ; 右下方圆角
LINEAR
X35Y50                               ; 右边
CIRCLE2
X30Y55I-5                            ; 右上方圆角
LINEAR
X10Y55                               ; 顶边
CIRCLE2
X5Y50J-5                             ; 左上方圆角
LINEAR
X5Y10                                ; 左边
```

```
CIRCLE2
X10Y5I5                    ; 左下方圆角
DWELL10                    ; 确保到达该位置
M1=0                       ; 关断输出
RAPID
X0Y0                       ; 返回原点
CLOSE
```

例36: 执行运动程序

这个程序演示了如何在8个不同坐标系中异步执行同一个电机运动程序.就好象在同一时间运行8个不同的程序，程序间彼此独立。一个PLC程序控制M1到M8八个变量。当这些变量中的任何几个被设置为1时，对应的坐标系执行程序，且独立于其它的坐标系。当这些变量中任何几个被设置为0时，对应的坐标系停止停止执行程序，且不影响其它坐标系。在本例中，M变量是自定义的，使得它们同P变量行为相同。但是，你可以定义这些M变量指向机器输出来切换读状态，这样可以开始和停止程序的执行。

```
CLOSE                      ; 关闭所有可能打开的缓冲器
UNDEFINE ALL               ; 清除所有当前的坐标系定义
&1                         ; 选择一号坐标系
#1->X                      ; 定义一号电机为X轴
&2                         ; 选择二号坐标系
#2->X                      ; 定义二号电机为X轴
&3                         ; 选择三号坐标系
#3->X                      ; 定义三号电机为X轴
&4                         ; 选择四号坐标系
#4->X                      ; 定义四号电机为X轴
&5                         ; 选择五号坐标系
#5->X                      ; 定义五号电机为X轴
&6                         ; 选择六号坐标系
#6->X                      ; 定义六号电机为X轴
&7                         ; 选择七号坐标系
#7->X                      ; 定义七号电机为X轴
&8                         ; 选择八号坐标系
#8->X                      ; 定义八号电机为X轴

M..8->*                    ; 规定M1到M8为自定义M变量
M1..8=0                    ; 令所有M1到M8变量为零
P1..8=0                    ; 设置运动/退出标志位为零

OPEN PLC 1 CLEAR           ; 打开PLC缓冲器1(注意:CS=坐标系)
IF(M1=1ANDP1=0)            ; 开关处于开状态且该坐标系当前没运行
  CMD“&1B1R”               ; 发出命令开始执行程序
  P1=1                     ; 使我们知道程序正在运行
ENDIF
IF(M1=0ANDP1=1)            ; 开关处于关状态且该坐标系当前正在运行
  CMD“&1Q”                 ; 发出命令停止程序
  P1=0                     ; 使我们知道程序已经停止运行
ENDIF
IF(M2=1ANDP2=0)            ; 开关处于开状态且该坐标系当前没运行
  CMD“&2B1R”               ; 发出命令开始执行程序
  P2=1                     ; 使我们知道程序正在运行
ENDIF
IF(M2=0ANDP2=1)            ; 开关处于关状态且该坐标系当前正在运行
  CMD“&2Q”                 ; 发出命令停止程序
  P2=0                     ; 使我们知道程序已经停止运行
ENDIF
IF(M3=1ANDP3=0)            ; 开关处于开状态且该坐标系当前没运行
  CMD“&3B1R”               ; 发出命令开始执行程序
```

```
   P3=1                                  ; 使我们知道程序正在运行
ENDIF
IF(M3=0ANDP3=1)                          ; 开关处于关状态且该坐标系当前正在运行
   CMD“&3Q”                              ; 发出命令停止程序
   P3=0                                  ; 使我们知道程序已经停止运行
ENDIF
IF(M4=1ANDP4=0)                          ; 开关处于开状态且该坐标系当前没运行
   CMD“&4B1R”                            ; 发出命令开始执行程序
   P4=1                                  ; 使我们知道程序正在运行
ENDIF
IF(M4=0ANDP4=1)                          ; 开关处于关状态且该坐标系当前正在运行
   CMD“&4Q”                              ; 发出命令停止程序
   P4=0                                  ; 使我们知道程序已经停止运行
ENDIF
IF(M5=1ANDP5=0)                          ; 开关处于开状态且该坐标系当前没运行
   CMD“&5B1R”                            ; 发出命令开始执行程序
   P5=1                                  ; 使我们知道程序正在运行
ENDIF
IF(M5=0ANDP5=1)                          ; 开关处于关状态且该坐标系当前正在运行
   CMD“&5Q”                              ; 发出命令停止程序
   P5=0                                  ; 使我们知道程序已经停止运行
ENDIF
IF(M6=1ANDP6=0)                          ; 开关处于开状态且该坐标系当前没运行
   CMD“&6B1R”                            ; 发出命令开始执行程序
   P6=1                                  ; 使我们知道程序正在运行
ENDIF
IF(M6=0ANDP6=1)                          ; 开关处于关状态且该坐标系当前正在运行
   CMD“&6Q”                              ; 发出命令停止程序
   P6=0                                  ; 使我们知道程序已经停止运行
ENDIF
IF(M7=1ANDP7=0)                          ; 开关处于开状态且该坐标系当前没运行
   CMD“&7B1R”                            ; 发出命令开始执行程序
   P7=1                                  ; 使我们知道程序正在运行
ENDIF
IF(M7=0ANDP7=1)                          ; 开关处于关状态且该坐标系当前正在运行
   CMD“&7Q”                              ; 发出命令停止程序
   P7=0                                  ; 使我们知道程序已经停止运行
ENDIF
IF(M8=1ANDP8=0)                          ; 开关处于开状态且该坐标系当前没运行
   CMD“&8B1R”                            ; 发出命令开始执行程序
   P8=1                                  ; 使我们知道程序正在运行
ENDIF
IF(M8=0ANDP8=1)                          ; 开关处于关状态且该坐标系当前正在运行
   CMD“&8Q”                              ; 发出命令停止程序
   P8=0                                  ; 使我们知道程序已经停止运行
ENDIF
CLOSE                                    ; 关闭PLC缓冲区

OPEN PROG 1 CLEAR                        ; 打开并清除程序缓冲器5
ABS                                      ; 设置绝对模式
TA500                                    ; 设置加速/减速时间为500毫秒
TS10                                     ; 用20毫秒时间进行S形曲线加速/减速
F10000                                   ; 设置进给速度10000计数脉冲每IX90毫秒
N10                                      ; 程序标号—后面要跳转到此处
X10000                                   ; 将X轴移动到坐标10000的位置
DWELL400                                 ; 停住400毫秒
X0                                       ; 回到坐标0位置
DWELL400                                 ; 再等待400毫秒
GOTO10                                   ; 重复!(直至PLC程序告诉我们停止)
```

```
CLOSE                                   ; 关闭运动程序缓冲区

I5=2                                    ; 允许程序PLC1工作
ENABLE PLC1                             ; 打开程序PLC1
```

例37: G代码编程

这个程序演示了在PMAC程序中G代码的使用。程序使用一小部分常用的G代码用三个轴切割部分材料(L形肋板)。在PMAC中，G代码实际是做为包含真正PMAC命令的子程序调用的。但是，
可以利用如同在机器,如CNC上使用G代码同样的方法使用它。它们作为子程序调用增加灵活性,使你可以用G代码完成任何需要的任务。

```
CLOSE                                   ; 关闭所有打开的缓冲区
OPEN PROG 6 CLEAR                       ; 打开并清除电机程序缓冲区6
G17                                     ; 选择画圆的平台
G20                                     ; 选择单位为英寸
G90                                     ; 选择绝对模式
G94                                     ; 选择英寸每分钟模式
F100                                    ; 设置进给速度为100英寸每分钟
G00 X1 Y4                               ; 迅速到达初始点
M3 S1800                                ; 开始轴以1800 RPM速度顺时针旋转
G01 Z.1                                 ; 降下切割刀具

M08                                     ; 释放冷却剂
Z0                                      ; 将切割刀具插入材料
Y12                                     ; 切割
G02 X2 Y13 I1 J0                        ; 切割圆角
G01 X3                                  ; 横向切割
G02 X4 Y12 I0 J-1                       ; 切割圆角
G01 Y6                                  ; 向下切割
G03 X7 Y4 I3 J0
G01 X13                                 ; 横向切割
G02 X14 Y3 I0 J-1
G01 Y2                                  ; 向下切割
G02 X13 Y1 I-1 J0
G01 X4                                  ; 向后横向切割
G02 X1 Y4 I0 J4                         ; 最后一个滚圆的边
G01 Z.1                                 ; 从材料处移走刀具
X3 Y3                                   ; 移动到下一点
Z0                                      ; 将切割刀具插入材料
G02  I.5  J.5                           ; 切割第一个洞
G01 Z.1                                 ; 从材料处移走刀具
X2 Y11                                  ; 移动到下一点
Z0                                      ; 将切割刀具插入材料
G02  I.5  J.5                           ; 切割第一个洞
G01  Z.1                                ; 从材料处移走刀具
X12  Y2                                 ; 移动到下一点
Z0                                      ; 将切割刀具插入材料
G02  I.5  J.5                           ; 切割第一个洞
G01  Z.1                                ; 从材料处移走刀具
M09                                     ; 关断冷却剂
Z1                                      ; 左切割刀具
G00 X0  Y0                              ; 返回原始位置
M05                                     ; 停止轴运动
CLOSE                                   ; 关闭程序缓冲区

; 下面是G代码程序中一些M变量定义
M163→D:$080B                            ; 电机1运动终点目标位置
```

```
M165→L:$081F                    ；坐标系1X轴运动终点目标位置
M191→L:$0822                    ；一号电机X轴系数
M194→L:$0825                    ；一号电机轴偏移量
M263→D:$080B                    ；电机2运动终点目标位置
M265→L:$081F                    ；坐标系1Y轴运动终点目标位置
M292→L:$0823                    ；二号电机Y轴系数
M294→L:$0824                    ；二号电机轴偏移量
M363→D:$098B                    ；电机3运动终点目标位置
M365→L:$0821                    ；坐标系1Z轴运动终点目标位置
M393→L:$09A4                    ；三号电机Z轴系数
M394→L:$09A5                    ；三号电机轴偏移量

OPEN PGOG 1000 CLEAR            ；G代码定义子程序
RAPID RET                       ；G00
N1000  LINEAR  RET              ；G01
N2000                           ；G02
CIRCLE1  RET                    ；顺时针圆弧

N3000                           ；G03
CIRCLE2 RET                     ；逆时针圆弧
N17000 NORMAL K-1               ；G17—定义X—Y平台
RET
N20000                          ；G20—英寸模式
M191=2000                       ；X轴比例因子为2000计数脉冲/英寸
M292=2000                       ；Y轴比例因子为2000计数脉冲/英寸
M393=2000                       ；Z轴比例因子为2000计数脉冲/英寸
P163=M163/(I108*32)             ；将一号电机位置换算为脉冲数
M165=(P163-M194)/M191           ；改变X轴位置
P263=M263/(I208*32)             ；将二号电机位置换算为脉冲数
M265=(P263-M294)/M292           ；改变Y轴位置
P363=M363/(I308*32)             ；将三号电机位置换算为脉冲数
M365=(P363-M394)/M393           ；改变Z轴位置
M70=0                           ；注意非公制的标志
RET
N90000 ABS                      ；G90—绝对模式
M90=1                           ；模式标志
RET
N94000  I190=60000              ；G94—进给速度是按每分钟
I193=2054                       ；使用内部时间机制
RET
CLOSE                           ；关闭程序缓冲区

OPEN PROG 1001 CLEAR            ；M代码定义子程序
N03000 READ (S)                 ；M03—打开转轴的方向信号
IF(Q100&262144>0)               ；如果S参数被定义
    I422=Q119/30                ；设置(转轴)慢进给速度
ENDIF
CMD“#4J+”                       ；四号电机慢进给驱动轴
RET
N05000                          ；M05—关断转轴
CMD“#4J/”                       ；停止转轴运动
RET N08000                      ；M08—释放冷却剂
M1=0                            ；机器输出=1
RET
N09000                          ；M09--停止释放冷却剂
M1=0                            ；机器输出=0
RET
CLOSE                           ；关闭程序缓冲区
```

例38: 程序延时执行

这个程序演示了如何在两个独立的坐标系中以一定的错位延迟运行同一个程序。在本例中，在坐标系1中的一号电机首先开始运动，二号电机接着运动,通过Q1定义滞后400毫秒。相同的Q变量在不同的坐标系中可以包含独特值，因此在PMAC中Q变量是特殊的。这里，坐标系1中Q1=0，坐标系2中Q1=400。并且，由于两个坐标系中运行同一个程序，因为Q2在坐标系1与坐标系2—
我们不应使用一个P变量既做延迟变量又做循环计数。因此它被用作循环计数。若要运行程序，输入：&1B1R&2B1R。

```
UNDEFINE  ALL                ；清除所有坐标系定义
&1                           ；选择坐标系1
#1->X                        ；定义一号电机为X轴
I190=1000                    ；设置进给速度时间单位为秒
Q1=0                         ；设置延迟变量为0毫秒
&2                           ；选择坐标系2
#2->X                        ；选择坐标系2
I290=1000                    ；设置进给速度时间单位为秒
Q1=400                       ；设置延迟变量为400毫秒

OPEN PROG 1 CLEAR            ；打开并清除程序缓冲器1
Q2=0                         ；将计数器清零
ABS                          ；设置绝对模式
TA100                        ；设置加速时间为100毫秒
TS0                          ；无S形曲线曲线时间
F10000                       ；设置进给速度为10000计数脉冲/秒
DELAY(Q1)                    ；延迟Q1中以毫秒规定的量
WHILE(Q2<10)                 ；开始一个10次的循环
   X2000                     ；X轴运动2000计数脉冲
   DELAY250                  ；延迟250毫秒
   X4000                     ；再运动2000计数脉冲
   DELAY250                  ；延迟250毫秒
   X6000                     ；再运动2000计数脉冲
   DELAY250                  ；延迟250毫秒
   X0                        ；返回0坐标位置
   DELAY250                  ；延迟250毫秒
   Q2=Q2+1                   ；计数器加1
ENDWHILE                     ；循环结束
CLOSE                        ；关闭程序缓冲区
```

例39: 积分电流极限

这个例子演示了如何使用PMAC中的PLC程序创建积分电流极限(I2T,“Eye-平方-Tee”保护)来支持或替代放大器。这种保护的目的,相对于瞬间电流极限,是试图避免放大器与/或电机的过热。如此,它使用的运动时间取于电流(在数字上等效于电容充电和放电)平方的平均。使用平方函数的目的是因为功率损耗与电流平方成正比。

这个例子使用命令电流输出
,因此只对电流模式放大器生效。一个更加健壮和灵活的版本将应用于测定的电流,但是这种情况要求有一个模数转换附件。但是,这两种情况下数学和逻辑是等效的。

```
；*****************************设置与定义*********************************************************************

CLOSE
M132→X:0045,8,16,S           ；一号电机伺服环16字节输出命令—在发送
                             ；到数模控制器之前.该寄存器的使用允许这
                             ；种技术,即使对PMAC的整流电机
P130=16384*16384             ；平方电流极限:16K*16K字节^2.
```

```
                                        ; 时间平均电流超过16K字节(最大值的一半)
                                        ; 程序将关断驱动器
P131=512                                ; PLC程序的数目扫描电流的平均值.
                                        ; 这个数与典型的PLC扫描周期时间相乘应
                                        ; 应该大约等于电机的热时间常数

; *************************实际的PLC程序*************************************************************

OPEN PROG 1 CLEAR`
P132=0                                  ; 确保平均值已被清除以便开始执行程序
DISABLE  PLC  1                         ; 这样在电源开/重置时只运转一次
CLOSE

OPEN PLC 5 CLEAR
P132=(P131-1)*P132+M132*M132)/P131      ; 计算运行平方的平均值
IF (P132>P130)                          ; 超越极限
     CMD“&1A”                           ; 停止坐标系中的所有运动
     CMD“#1K”                           ; 关断电机
     SEND“Motor 1 I2T Limit  Trapped”   ; 传送信息到主机
     M1=1                               ; 打开指示器
     P132=0                             ; 清除运行平均值
ENDIF
CLOSE
```

II 主程序示例

PMACPOLL.C

```
/*示例程序中使用查询准备位
*/

#include <stdio.h>
#include <dos.h>
#include <conio.h>

#define   SERIAL          555
#define   PCBUS           123
#define   COM1            1016
#define   COM2            760
#define   PMAC-PIC02      528+10
#define   PMAC-PIC03      528+12

int      combase,                       /*卡的基址*/
         baudrate,                      /*串行通讯波特率*/
         speed,                         /*用于计算超时*/
         able_to_talk,                  /*确认通讯的标志*/
         timeout;                       /*查看信号交换输入输出间隔时间限制*/
char     buf[256];                      /*保持输入字符串*/

int  config_card_for (int port,int address){
```

/*时钟速度乘法器设置查询信号交换的时间极限,以进行读写操作。在结束运行之前,软件要查看为正确的信号交换所输入的数字700次。CPU的速度越快,则这个值越大。对PMAC来说,完成这个控制所要决定的唯一事情就是多长时间检查是否PMAC准备发送一个线道。
对波特率的设置,使用下列值：

波特率期望值	波特率变量使用值
300	波特率= 384
1200	波特率= 96
2400	波特率= 48
4800	波特率= 24
9600	波特率= 12
19200	波特率= 6
38400	波特率= 3

```
确保你已经将E触点跳线连接为匹配你的波特率形式!*/
        speed=9;
        combase=address;
        if (port==SERIAL) {
                baudrate=12;                        /*设置9600波特率
                timeout=7*speed*(baudrate+100);
                outportb(combase+3,131);            /*设置PC串口进行通讯*/

                outportb(combase,baudrate);
                outportb(combase+1,baudrate/256);
                outportb(combase+3,3);
        }
        else {                                      /*PC总线被用于通讯*/
                baudrate=0;                         /*当波特率设为0时,PC总线通讯被采用*/
                timeout=7*speed*100;
                outportb (combase+5,0);
                outportb (combase+6,0);
        }
}

int sendline (char outchar) {                       /*调用函数实现在CR后发送一串字符到
                                                    PMAC*/
        int i=0;
        while (outchar [i]!=0)
                sendchar (outchar [i++]);
        sendchar (13);
}
int sendchar (char outchar) {                       /*发送一个字符到PMAC-PC卡时注意:即使是
                                                    25MHZ的386PC机也不能“超过”PMAC
                                                    运 行速度,因此查看写准备字节可能是多余的
                                                    */

        int i;
        i=0;
        if (baudrate= =0) {                         /*通过PC总线发送数据*/
                                                    /*查询写准备好节(基址加2地址的第1位)*/
                while (i++<timeout && !(intportb (combase+2)&2));
                if (i<timeout)
                        ourportb(combase+7,outchar);
        else {                                      /*由RS232接口(串口)传送数据*/
                                                    /*查询写准备字节(基址+5&6的第6位&第5
                                                      位)*/
                while(i++<timeout && !(intportb (combase+5)& 32 )= =0);
```

```
                while(i++<timeout && !(intportb (combase+6)& 16 )= =0);
                if (i<timeout)
                        outportb(combase,outchar);

        }
        return (0);
}

int getline (char *linebuf) {
/*查询来自PMAC-PC卡的一行字符数据
注意：PMAC卡运行足够快，因此可能只需在行开始检查读准备好字节（事先进行超时设定）--
一旦开始读这一行，该状态位的检查就可能是多余的*/

        char ic;
        int i,     nc;

        if(able_to_talk) {
                ic=nc=i=0;
                if (baudrate= =0){                              /*通过PC总线接口读数据*/
                        while (i++< timeout && ic!=13 &&nc<255)
/*查询读准备好字节(基址加2地址的第0位)*/
                        if (( inportb (combase +2 ) &1 )= =1) {
/*发现它..读一个字符并加到字符串末尾*/
                                ic=inportb (combase+7);
                                linebuf [nc++]=ic;
                                i=0;
                        }
                }
                else {                                          /*通过RS232接口读数据*/
                        outportb (combase+4,2);     /*解除“hold off”信号*/
                        while (i++<timeout && ic!=13 && nc<255){
                                if(( inportb (combase +5)&1)= =1){
                                        ic=inportb (combase);
                                        enable ( );
                                        linebuf [nc++]=ic;
                                        i=0;
                                        disable ();
                                        }
                        }
                        outportb (combase+4,0);    /*设置“hold off”信号*/
                        enable ();
                        while ((i+=2)<timeout);
                }
                linebuf [nc] =0;
                return (nc);                                    /*返回接收到的字符个数*/
        }
        else {
                linebuf [0]=0;
                return (0);                                     /*没有接收字符*/
        }
    }

int Cards_on_Line (){                                           /*查证PMAC是否在位*/
        int  i,n;
        able_to_talk=1;
        n=i =0;
        while (n<3 && i<11){
                if (baudrate !=0){                              /*仅为串行通讯*/
                        sendchar (24);                          /*发送^X*/
```

```
                    delay (20);
                    sendchar (26);              /*为串行模式发送^Z*/
                }
                sendline ("RHL:$720");
                i = getline (buf);
                n++ ;
        }
        if  (i>11 && i<17) {
                cprintf ("Found PMAC…\r\n");
                return (1);
        }
        else{
                sound (1000);  delay (250);  nosound ();
                cprintf ("No  PMAC Found !\r\n");
                ble_to_talk = 0;
                return (0);
        }
}

void show_printf(char *lbuf){
int i;
for(i=0; I<strlen(lbuf); I++){
        if(lbuf[I]==13/*||lbuf[I]==7*|| lbuf[I]==6||lbuf[I]==10)
                switch  (lbuf[I]){
                        case 13:printf("<CR>\r\n"); break;
                        case 7:printf("<BELL>\r\n"); break;
                        case 6: printf("<ACK>"); break;
                        case 10:printf("<lF>\r\n"); break;
                }
        else{
                if (isprint (lbuf[I]))
                        putch(lbuf[I]);
                else
                        printf("<%u>",lbuf[I]&255);
        }
}
}

void main(){
        int done=0;
        char ic;
//config_card_for(PCBUS,528);                   //使用这行来通过PC总线通话!
 Config_card_for (SERIAL,COM1);
Cards_on_line ();
cprintf("Now acting as a terminal to PMAC.\r\n");
cprintf("press <ESC> to abort.\r\n");
while(!done && able_to_talk){
        while(!kbhit()) {                       /*没有键按下,因此检查卡上的数据*/
                getline (buf);
//              if (show_codes)
//                      show_printf(buf);
//              else
                        printf (buf);
        }
        switch (ic=getch ()) {
                case 1:
                  outp(PMAC_PIC03,0x40);
                  delay(4000);
                  outp(PMAC_PIC02,0x40);
                  break;
```

```
                case 27:                        /*按下ESC键,则退出*/
                done=1；break；
              case 0:                           /* 接收第2部分特殊键(例如F键)
                getch()；break；
              default:
                sendchar(ic)；                  /*发送字符到PMAC*/
                if (ic= =13)                    /*按下ENTER键*/
                      putch(10)；
                putch(ic)；
                break；
              }
        }
}
```

PMACINT.C

/*示例程序使用中断同PMAC通话。确保关于IRQ5的跳线E83被设置允许使用PC的中断IRQ5.*/

```
/*-----------------------PMAC总线端口地址----------------------*/
#define PMACBUFFERSIZE          1024                    /*这是队列缓冲区的尺寸*/
#define  PMAC                   528                     /*PMAC总线的地址*/
#define  PMAC_STATUS            PMAC+2                  /*总线状态字节*/
#define  PMAC_DATA              PMAC+7                  /*传送/接收数据*/
#define  PMAC_PIC00             PMAC+8                  /*8259中断控制器*/
#define  PMAC_PIC01             PMAC+9                  /*      “      ”      ”*/
#define  PMAC_PIC02             PMAC+10                 /*      “      ”      ”*/
#define  PMAC_RECEIVE           2                       /*准备接收位*/
#define  PMAC_SEND              1                       /*准备发送位*/

#define  PMAC_IS_SENDING  (inp  (PMAC_STAUS)  &  PMAC_SEND)
/*-----------------------PMAC communication  interrupt  values---------------*/

#define  PMAC_IRQ               4                       /*0-7  IRQ0-IRQ7*/
#define  PMAC_MASK              (~(1<<PMAC_IRQ))        /*去掉PMAC 8259的屏蔽*/
#define  PMAC_EOI               0x20                    /*PMAC中断停止*/
#define  PMAC_IPOS              1                       /*IR0中断用于位置*/
#define  PMAC_BREQ              2                       /*IR1用于缓冲区请求*/
#define  PMAC_ERROR             4                       /*IR2用于一般错误*/
#define  PMAC_FERROR            8                       /*IR3用于跟随错误*/
#define  PMAC_HREQ              16                      /*IR4用于通讯*/
#define  PMAC_IR5               32                      /*IR5*/
#define  PMAC_IR6               64                      /*IR6*/
#define  PMAC_IR7               128                     /*IR7*/
#define  PMAC_NOP               0x40                    /*8259没有操作代码*/

/*--------------------PC中断值----------------------*/

#define    IRQ                  5                       //PC用于PMAC的中断:0-15对应
                                                        //于IRQ0-IRQ15
# if (IRQ<8)                                            //对XT/AT的中断等级
//#define  PC_INT               (8+IRQ)
#define    PC_INT               0x77
#define    PC_MASK              (~(1<<IRQ))             //去掉PC机8259的屏蔽
#define    PC_PIC01             0x21                    //8259的地址(AT总线中的主导PIC)
#define    PC_PIC00
#else                                                   //仅为AT总线的中断向量
#define     PC_INT              (104+IRQ)
#define     PC_MASK             (~(1<<(IRQ-8)))         //去掉PC机8259的屏蔽
```

```
#define     PC_PIC01            0xA1                    //第二个8259的地址(AT总线中从属PIC)
#define     PC_PIC00
#endif

#define      PC_EOI          0x20                       //中断结束

#define     ACK                 0x06
#define     LF                  0x0c
#define     EOL                 0x0d

#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <bios.h>

static  void       (interrupt  far  *old_comm_int)  ();
static  void       interrupt  far  pmac_comm();

static  unsigned          combase,speed,able_to_talk,timeout,old_int_stat;
static  char              ipos_flag,breg_flag,error_flag,ferror_flag,hreg_flag,
                  ir5_flag,buf[256],comm_buffer[PMACBUFFERSIZE],
                  current_buffer,next_in,*next_out;

void init_pmac_comm () {                        /*为中断通讯初始化PMAC端口*/

   setup_pmac_queue    (comm_buffer);           /*设置输入PMAC队列*/
   disable ();                                  /*关闭中断直至设置结束*/
   old_comm_int=getvect       (PC_INT);         /*保存原始中断变量*/
   setvect   (PC_INT,pmac_comm);                /*写新的中断变量!*/

/*----------------设置PC机的8259…--------------------*/
    /old_int_stat = inp   (PC_PIC01);
    outp   (PC_PIC01,(old_int_stat  &   PC_MASK));

/*----------------现在设置PMAC卡的8259…--------------------*/

    outp  (PMAC,0);                             /*冲掉PMAC中断*/
    outp  (PMAC_PIC00,0x17);                    /*0x17=edge,0xif=级连触发 (ICW1)*/
    outp  (PMAC_PIC01,0x08);                    /*数据总线向量(ICW2)*/
    outp  (PMAC_PIC01,0x03);                    /*设置8086模式(ICW4)*/
    outp  (PMAC_PIC01,0x00);                    /*屏蔽掉IR6,IR7中断*/

/*如果想使用所有的中断输入,请使用0x00代替0xE0*/
   outp(PMAC,0x81);                             /*DSP允许读*/
   enable();                                    /*设置结束,打开中断*/
}
void restore_pmac_comm  ()  {                   /*保存原始中断向量*/

    disable ( ) ;
setvect  (PC_INT,old_comm_int);                 /*取代原有的PC向量*/
outp  (PC_PIC01,old_int_stat);                  /*保存原有的状态*/
outp  (PMAC_PIC01,0xff);                        /*关断PMAC的8259*/
enable  ( ) ;
}

static  void  interrupt  far pmac_comm  (void)  { /*  PMAC中断服务子程序*/

    int        ch;
    char       isr;
```

```
    sound  (3000)；delay (10)；  nosound ()；          /*中断使蜂鸣器响*/
    disable  ()；                                      /*直至结束不中断*/
    outp (PMAC_PIC02,PMAC_NOP)；                       /*以不同的INTA应答中断*/
    outp (PMAC_PIC00,0x0A)；                           /*设置读IRR寄存器*/
    isr = inp(PMAC_PIC00)；                            /*读IRR寄存器*/
    if (isr & PMAC_IPOS)                               /*如果设置IPOS中断,建立标志*/
        ipos_flag=1；
    if (isr& PMAC_BREQ)                                /*如果设置BREQ中断,建立中断*/
        breq_flag=1；
    if (isr & PMAC_ERROR)                              /*如果设置ERROR中断,建立中断*/
        error_flag=1；
    if (isr & PMAC_FERROR)                             /*如果设置FERROR中断,建立中断*/
        ferror_flag=1；
    if (isr & PMAC_HERQ)                               /*如果设置HREQ中断,建立中断*/
        hreq_flag=1；
//  if(isr & PMAC_IR5){                                /*如果设置IR5中断,建立中断*/
        ir5_flag=1；
//        sound (3000)；delay (10)；nosound ()；
// }
        outp(PMAC_PIC02,PMAC_NOP)；                    /*再次应答中断*/
        outp(PMAC_PIC00,PMAC_EOI)；                    /*向PMAC的8259发送中断结束信息*/
        outp (PC_PIC00,PC_EOI)；                       /*向PC的8259发送中断结束信息*/

        enable()；                                     /*再次开中断*/
        (*old_comm_int)  ( ) ；                        /*跳转到替换子程序*/
}
int  setup_pmac_queue  (char  *buf)      {
        next_in =next_out =current_buffer =buf；
        return  (0) ；
}
int read_pmac_queue (char buf)                          { /*为PMAC响应设立的循环队列缓冲器*/

        int nc=0, done=0；
        while  ( !done && ( next_in !=next_out ) )        {
            if(next_out==current_buffer+PMACBUFFERSIZE)
                next_out=current_buffer；
                if  ((*next_out==EOL)||(*next_out==ACK)){
                  next_out++；
                  done=1；
        }
                else
                  buf[nc++]=*next_out++；
}
buf[nc]=0；
return (nc)；
}

void config_card_for (unsigned address) {
speed=9；
combase=address；
timeout=7*speed*100；
outportb(combase+5,0)；
outportb(combase+6,0)；
}

void sendchar  (char outchar) {
int i=0；
while (i++ <timeout &&!(inportb(combase+2) & 2))；
if(I<timeout)
```

```
        outportb(combase +7,outchar);
}
void sendline (char *outchar){
int i=0;
while(outchar[i]!=0)
        sendchar(outchar[i++]);
sendchar(13);
}

int getline (char *linebuf) {
char ic;
int i,nc;
if(able_to_talk) {
        ic=nc=i=0;
        while (i++<timeout && ic!=13 && nc<255)
                if((inportb (combase+2) &1)==1){
                        ic=inportb(combase +7);
                        linebuf[nc++]=ic;
                        i=0;
                }
        linebuf[nc]=0;
        return (nc);                    /*PMAC要信息告知*/
}
else {
        linebuf[0]=0;
        return (0);                     /*PMAC无信息告知*/
}
}

int Cards_on_Line () {
int i,n;
sendchar (24);                          //发送^X
delay (500);
able_to_talk=1;
n=i=0;
while (n<3 && i<11){
        sendline("RHL:$720");
        i=getline(buf);
        n++;
}
if (i>11 && i<17) {
        cprintf("\r\nFOUND  PMAC…\r\n");
        while(getline (buf));           /*清空PMAC的输出缓存器*/
`       return (1);
}
else {
        sound (1000); delay(250); nosound ();
        cprintf ("\r\nNo PMAC found!\r\n");
        able_to_talk=0;
        return (0);
}
}
void  do_interrupt_terminal (){
int key,done=0,pos;
char str[255],ch;

init_pmac_comm ();
sendline ("i3=0");
while (!done){
/*  现在处理中断*/
```

```
        if(hreq_flag) {                                 /*PMAC是否有信息通知?*/
                hreq_flag=0;
                while (PMAC_IS_SENDING){                /*实际上为查询读数据*/
                        if(nest_in==current_buffer+PMACBUFFERSIZE)
                        next_in=current_buffer;         /*读PMAC并存贮数据到队列中*/
                        *next_in++=(char) inp (PMAC_DATA);
                }
        }
        if(ferror_flag) {
                ferror_flag=0;
                cprintf ("\r\n***Following Error!***\r\n");
        }
        if(error_flag) {
                error_flag=0;
                cprintf("\r\n***General Error!***\r\n");
        }
        if(breq_flag) {
                breq_flag=0;
                cprintf("\r\n***Buffer Request!***\r\n");
        }
        if(ipos_flag) {
                cprintf("\r\n***In Position**%x\r\n",(int) ipos_flag) ;
                ipos_flag=0;
                }

*包含此代码段来实现每个附加打开的中断*/
/*      if(ir5_flag) {
                cprintf("\r\n***Interrupt IR5 %x***\r\n",(int) ir5_flag);
sendline("p"); getline(buf)l
sscanf(buf+1,"%d",&pos);
printf("%d%d\r\n",pos,pos%360);
                ir5_flag=0;
        }
*/  if(kbhit()) {
        key=getche ();                  /*从键盘接收一个字符*/
        switch (key) {
                case 27:                /*按下ESC键,退出*/
                        done=1; break;
                case 13:                /*按下ENTER键,打印LF*/
                        cprintf("\n");
                default:
                        sendchar (key);
        }
    }
    if(read_pmac_queue (str)!=0)
        puts(str);                      /*打印队列中所有内容*/
}
restore_pmac_comm();
sendline("i3=1");
}

void main () {
clrscr;
config_card_for(0xF090);
// config_card_for(528);
cards_On_Line();
ipos_flag=breq_flag=error_flag=ferror_flag=hreq_flag=ir5_flag=0;
if(able_to_talk){
        cprintf("Make sure a jumper is installed on E80.\r\n");
```

```
        cprintf('Now acting as an interrupt driven terminal to PMAC.\r\n");
        cprintf("press <ESC> to abort.\r\n");
        do_interrupt_terminal ();
}
}
```

PMACPROT.C

```
/*PMAC查询循环存储转储程序.*/
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <dir.h>

#define SERIAL          321
#define PCBUS           123
#define COM1  1026
#define COM2  760
#defineTRUE   1
#define FALSE 0
FILE            *tempfile;
Int             combase,baudrate,speed,able_to_talk,timeout;
char            buf[256];
struct          ffblk ffblk;

int  config_card_for (int port,int address) {
speed=9;
combase=address;
if (port==SERIAL) {
        baudrate=12;                    /*选择9600波特率*/
        timeout=7*speed*(baudrate+100);
        outportb (combase +3,131);      /*设置PC用于通讯的串行口*/
        outportb (combase,baudrate);
        outportb (combase+1,baudrate/256);
        outportb (combase+3,3);
}
else {                                  /*PC总线将被用于通讯*/
        baudrate=0;                     /*当波特率被设置为0时,PC总线通讯采用*/
        timeout=7*speed*100;
        outportb(combase+5,0);
        outportb(combase+6,0);
}
}
int sendline (char outchar) {           /*调用函数可以在CR后发送一串字符到 PMAC*/
    Int i=0;
    while (outchar[i]!=0)
        sendchar (outchar[i++]);
    sendchar (13);
}
int sendchar (char outchar) {           /*发送一个字符到PMAC*/
int i;
i=0;
if(baudrate==0) {                       /*通过PC总线接口发送字符*/
        while(i++<timeout && !(inportb(combase+2)&2));
        if (i<timeout)
                outportb(combase+7,outchar);
}
else{                                   /*通过RS232(串口)接口发送数据*/
        while (i++<timeout &&(inportb(combase+5)&32)==0);
```

```
        while (i++<timeout &&(inportb(combase+6)&16)==0);
        if(i<timeout)
                outportb(combase,outchar);
}
return (0);
}

int getline (char *linebuf){
char ic;
int i,nc;
if(able_to_talk) {
 ic=nc=i=0;
 if (baudrate==0) {                    /*通过PC总线接口读数据*/
        while (i++ <timeout && ic!=13 && nc<255)
                if ((inportb (combase +2) &1)==1) {
                        ic=inportb (combase +7);
                        linebuf [nc++] =ic;
                        i=0;
                }
}
else {                                  /*从RS-232接口读数据*/
        outportb (combase+4,2);
        while (i++<timeout && ic!=13 && nc<255){
                if ((inportb (combase +5)&1)==1){
                        ic=inportb (combase);
                        enable ();
                        linebuf[nc++]=ic;
                        i=0;
                        disable();     }
        }
        outportb(combase+4,0);
        enable();
        while((i+=2)<timeout);
}
linebuf[nc]=0;
return(nc);                             /*返回接收字符的个数*/
}
else {
        linebuf[0]=0;
        return(0);                              /*未接收到数据*/
}
}
int getchare () {                               /*优化过的getline子函数,高速下载数据到PMAC*/
char ic;
int  i,nc;
if(able_to_talk) {
        ic=nc=i=0;
        if (baudrate==0) {                      /*不是RS-232*/
                while (nc<1)
                        if((inportb (combase+2)&1)==1){
                                ic=inportb(combase+7);
                                nc++;
                        }
        }
        else {                                  /*RS-232*/
                outportb(combase+4,2);
                while (nc<1) {
                        if ((inportb(combase+5)&1)==1) {
                                ic=inportb(combase);
                                enable();
```

```
                                nc++;
                                i=0;
                                disable();    }
                        }
                        outportb (combase+4,0);
                        enable();
                        while ((i+=2) <timeout);
                }
                if (ic==7)
                        return (1);
                else
                        return (0);
}else
                return(0);
}
int Cards_On_Line (){
int  i,n,repeat;
able_to_talk=1;
for(repeat=1;  repeat<=3;  repeat++) {
                n=i=0;
                while (n<3 &&n<11){
                        if (baudrate >0)  {                /* 仅对串行通讯*/
                           sendchar (24);                  /*发送^X*/
                          delay (20);
                           sendchar (26);                  /*为串行模式发送^Z*/
                        }
                        sendline ("RHL:$720");
                        i=getline (buf);
                        n++;
                }
}
if (i>11 && i,17) {
                cprintf ("Found PMAC…\r\n");
                while (getline (buf));                  /*清空 PMAC的输出缓存器*/
                return (1);
}
else {
                sound (1000);  delay (250);   nosound ();
                cprintf("NO PMAC found!\r\n");
                able_to_talk=0;
                return (0);
}
}
int  lines_in_buffer () {                  /*发送PR命令到PMAC,PMAC在命令执行之
                                             前返回旋转寄存器中行的数目*/
        int lines_remaining;
        sendline("PR");
        getline (buf);
        sscanf (buf, "%d",& lines_remaining);
        return (lines_remaining);
}
int main (int argc,char *argv[]){
int      byte,i,aborted,done,key;
long   line;
char   filename[80],line_buf [256];
cprintf ("\r\n\nPMAC Rotary File Program\r\n");
/*config_card_for(PCBUS,528);  */

config_card_for (SERIAL,COM1);
```

```
Cards_On_Line ();
If ((argv[1][0]>=''&& argv[1][0]<='z') && able_to_talk) {
        sendline ("Q CLOSE I9=0 I3=1");
        sendline ("&1");                                    /*选择坐标系1*/
        sendline ("DELETE TRACE DELETE GATHER");
        sendline ("DEFINE ROT 250 CLEAR");          /*设置旋转寄存器*/
        while (getline (buf));
        strcpy (filename,argv[1]);
        done=aborted=FALSE;
        while (!aborted && !done) {
                if ((tempfile=fopen (filename,"rt")) !=NULL) {
                        findfirst (filename,&ffblk,0);      /*打开文件转存到旋转寄存器*/
                        i=line_buf[0]=0;
                        line=1;
                        while (fscanf (tempfile,"%c",&byte)!=EOF && !aborted){
                                if (kbhit())
                                if (getch()==27)            /*按下ESC键退出转存*/
                                        aborted=TRUE;
                                byte=byte&255;              /* 屏蔽高位字节*/
                                if (byte==10) {             /* 到达行末尾*/
                                while ((lines_in_buffer ()>250)&&((line-1)%250)<2 && !aborted);
                                if(kbhit())
                                if (getch()==27)            /*按下ESC键退出缓存*/
                                  aborted=TRUE;
                                  sendline (line_buf);  getchare();
                                        if (line>250)
                                                cprintf("*sending line:%ld\r",line);
                                        else
                                                cprintf("sending line:%ld\r",line);
                                        i=line_buf[0]=0;
                                        line++;
                                        if (line==250) {/*只在第一次时询问*/
                                                cprintf("\n\n\npress <ENTER> to
                                                eginexecution:\r");
                                                key=getch();
                                                while (key!=13 &&key !=27)
                                                   key=getch();  /*等待ESC或ENTER*/
                                                if (key==13)
                                                   sendline ("R");
                                                else {
                                                   sendline ("CLOSE");
                                                   aborted=TRUE;
                                                }
                                        }
                                }
                                else {                              /*因为还没到行的末尾,加到字符串后*/
                                        line_buf[I++]=byte;
                                        line_buf[i]=0;
                                }
                        }
                        done=TRUE;
                        if (line<250) {                             /*仅在第一次询问*/
                                cprintf("press <ENTER> to begin execution:");
                                key=getch();
                                while (key!=13 && key !=27)
                                        key=getch();
                                if (key==13)
                                        sendline ("close R");  /*如果是小文件,发布R*/
                                else
```

```
                                        aborted=TRUE；
                        }
                        if (!aborted)
                                cprintf(“\r\nDone.\r\n”)；
                        else {
                                cprintf(“\r\nAborted.\r\n”)；
                                sendline (“CLOSE”)；
                        }
                }
                else {
                        aborted=TRUE；
                        sound(1000)； delay(300)； nosound()；
                        cprintf(“\r\nUnable to open file %s”,filename)；
                }
        }
        fclose (tempfile)；
        sendline (“CLOSE”)； /*关掉转存缓冲器*/
        }
}
```

PMACIROT.C

/*PMAC中断驱动转存缓存器程序。
设置与IRQ5有关的跳线E83为允许PMAC向PC机相应IRQ5的中断。

*/

```
/*----------------------PMAC总线端口地址--------------------------*/
#define  PMACBUFFERSIZE  1024          /*队列缓存器尺寸*/
#define  PMAC                528       /*PMAC的总线地址*/
#define  PMAC_STATUS         PMAC+2    /*总线状态字节*/
#define  PMAC_DATA           PMAC+7    /*传送/接收数据*/
#define  PMAC_PIC00          PMAC+8    /*8259中断控制器*/
#define  PMAC_PIC01          PMAC+9    /*      “  ”      “  */
#define  PMAC_PIC02          PMAC+10   /*      “  ”      “  */
#define  PMAC_RECEIVE        2         /*准备接收位*/
#define  PMAC_SEND           1         /*准备发送位*/

#define  PMAC_IS_SENDING  (inp (PMAC_STATUS)&PMAC_SEND)

/*-----------------------------PMAC通讯中断值-------------------------*/

#define  PMAC_IRQ            4         /*0-7  IRQ0-IRQ7*/
#define  PMAC_MASK (~(1<<PMAC_IRQ))    /*去掉PMAC8259的屏蔽*/
#define  PMAC_EOI            0x20      /*PMAC中断结束*/
#define  PMAC_IPOS           1         /*位置中断IR0*/
#define  PMAC_BREQ           2         /*缓存器请求中断IR1*/
#define  PMAC_ERROR          4         /*一般性错误申请IR2*/
#define  PMAC_FERROR         8         /*跟随错误申请IR3*/
#define  PMAC_HREQ           16        /*通讯申请IR4*/
#define  PMAC_EQU1           32        /*DP RAM 应答请求IR5*/
#define  PMAC_NOP            0x40      /*8259无操作码*/

/*----------------------PC 中断值-------------------------*/

#define  IRQ            5              /*PC机用于PMAC的中断*/
#define  PC_INT         (8+IRQ)
#define  PC_MASK        (~(1<<IRQ))    /*屏蔽PC的8259*/
#define  PC_PIC01       0x21           /*8259中断控制器XT*/
```

```
#define  PC_PIC00     0x20                    /*“0-7”中断*/
#define  PC_EOI       0x20                    /*中断结束*/

#define  ACK          0x06
#define  LF           0x0C
#define  EOL          0x0D
#define  PCBUS        123
#define  TRUE         1
#define  FALSE        0

#include  <stdio.h>
#include  <dos.h>
#include  <conio.h>
#include  <stdlib.h>

static  void      (interrupt  far  *old_comm_int)  ();
static  void      interrupt  far  pmac_comm();

static  int  combase,speed,able_to_talk,timeout,old_int_stat,breq_flag;
static  long  x,y,z;
static  char  buf[256];

void init_pmac_comm () {                      /*为中断通讯初始化PMAC端口*/

        disable();                            /*关中断直至设置好*/
        old_comm_int=getvect(PC_INT);         /*保存原始中断向量*/
        setvect (PC_INT,  pmac_comm);         /*写入新的中断变量*/

/*------设置PC的8259------------*/
        old_int_stat=inp(PC_PIC01);
        outp(PC_PIC01,(old_int_stat & PC_MASK));

/*------现在设置PMAC的8259…---------*/
        outp(PMAC,0);                        /*去除PMAC中断*/
        outp(PMAC_PIC00,0x17);                /*0x17=edge,0xlf=级连触发器(ICW1)*/
        outp(PMAC_PIC01,0x08);                /*数据线向量    (ICW2)*/
        outp(PMAC_PIC01,0x03);                /*设置8086模式          (ICW4)*/
        outp(PMAC_PIC01,0xFD);
        outp(PMAC     ,0x81);                 /*允许读  DSP*/
        enable();                             /*设置结束,因此开中断*/
        breq_flag=0;                          /*清除中断标志*/
}

void restore_pmac_comm () {                   /*保存原始中断向量*/
        disable();
        setvect( PC_INT,old_comm_int);        /*替换旧的中断向量*/
        outp(PC_PIC01,old_int_stat);          /*保存旧的状态*/
        outp(PMAC_PIC01,0xff);                /*关断PMAC的8259*/
        enable();
}
static  void interrupt far pmac_comm(void) {     /*PMAC中断服务子程序*/
        int ch;
        char isr;
/*sound(3000);  delay(10);  nosound();          /*当中断时蜂鸣器响*/
        outp(PMAC_PIC02,PMAC_NOP);            /*提高第一个INTA/脉冲的沿*/
        outp(PMAC_PIC00,PMAC_NOP);            /*跟踪第一个INTA/脉冲的沿*/
        outp(PMAC_PIC02,0x0B);                /*设置读ISR寄存器*/
        isr=inp (PMAC_PIC00);                 /*读ISR寄存器*/
        if (isr & PMAC_BREQ)                  /*如果设置了BREQ中断,设一个标志*/
```

```
                breq_flag=1;
        outp(PMAC_PIC00,PMAC_NOP);      /*跟踪第二个INTA/脉冲的沿*/
        outp(PC_PIC00,PC_EOI);          /*发送中断结束信号到PC的8259*/
        enable();                       /*重新开中断*/
}

void config_card_for (int address) {
        speed=9;
        combase=address;
        timeout=7*speed*100;
        outportb(combase+5,0);
        outportb(combase+6,0);
}
void sendchar (char outchar) {          /*发送一个字符到PMAC*/
        int i=0;
        while(i++<timeout && !(inportb(combase+2) &2));
        if (i<timeout)
                outportb(combase+7,outchar);
}
void sendline (char outchar) {          /*发送一串字符和CR到PMAC*/
        int i=0;
        while (outchar[i]!=0)
                sendchar (outcahr [i++]);
        sendchar(13);
}
int getline (char linebuf) {            /*查询读PMAC*/
char ic;
int  i,nc;
if(able_to_talk) {
        ic=nc=i=0;
        while (i++<timeout && ic!=13 &&nc<255)
          if ((inportb(combase+2) & 1)==1) {
                ic=inportb(combase+7);
                linebuf[nc++]=ic;
                i=0;
          }
        linebuf[nc]=0;
        return(nc);                     /*PMAC有信息提供*/
}
else{
        linebuf[0]=0;
        return(0);                      /*PMAC没有信息提供*/
}
}
int Cards_On_Line() {                   /*检验PMAC在位*/
        int i,n;
        able_to_talk=1;
        n=i=0;
        while (n<3 && i<11) {
                sendline ("RHL:$720");
                i=getline(buf);
                n++;
        }
        if (i>11&& i<17){
                cprintf("\r\nFOUNDPMAC…\r\n");
                while (getline (buf));  /*清除PMAC的输出缓存器*/
                return (1);
        }
        else {
```

```
                sound (1000)；  delay(250)；l nosound()；
                cprintf (“\r\nNO PMAC found!\r\n”)；
                able_to_talk=0；
                return (0)；
        }
}

void send_command_to_PMAC () {                  /*将计算好的位置发送到PMAC*/
        char  command[80]；
        sprintf (command,“X%ld Y%ld Z%ld”,xyz)；
        sendline (command)；                    /*发送上次计算好的运动*/
        cprintf( “Sending X%ld Y%ld Z%ld\r”,x,y,z)；
        x+=2000；                               /*计算下次运动的位置*/
        y+=2000；
        z+=2000；
    }

void main () {

int done=FALSE；
char str[256]；
config_card_for (528)；                         /*在528为PC总线设置*/
Cards_On_Line ()；                              /*检验卡在位*/
if (able_to_talk){
        cprintf(“Now downloading positions to rotary…\r\n\n”)；
        sendline (“Q I16=1 I17=2”)；            /*设置旋转寄存器调用参数*/
        sendline (“I3=0”)；
        sendline (“#1hm#3hm#4hm”)；
        init_pmac_comm()；                      /*为子程序设置中断向量*/
        sendline (“DEFINE ROT 50”)；            /*为旋转寄存器分配空间*/
        sendline (“OPEN ROT”)；                 /*打开该寄存器*/
        sendline (“F20000”)；
        sendline (“LINEAR”)；
        while (getline (buf))
        sendline (“R”)；                        /*开始运行它*/
        while (!done) {
/*现在处理中断*/
                if(breq_flag) {                 /*如果设置了BREQ中断*/
                   breq_flag=0；                /*清除标志*/
                   sendline_command_to_PMAC ()；   /*下载一个计算好的运动
                                                   到PMAC*/
                }
/*后台任务可以插入此处*/
                if (kbhit())
                        if(getch()==27)         /*按下ESC键退出程序*/
                                done=TRUE；
        }
        sendline (“CLOSE Q”)；                  /*因设置完备,关掉旋转寄存器*/
        sendline (“I3=1”)；
        restore_pmac_comm()；                   /*保存原始中断设置*/
    }
    while (getline (buf))；
}
```

PMACLERN.C

/*示例程序给出了PMAC中的学习模式。该程序 (PROG123) 是实现运动的程序,每当按下CTRL-B时将产生程序中包含的一系列X轴运动*/

```
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#define SERIAL  555
#define PCBUS   123
#define COM1    1016
#define COM2    760

int     combase,                        //卡的基址
        baudrate,                       //串行通讯波特率
        speed,                          //用于计算超时
        able_to_talk;                   //确定通讯的标志
        timeout;                        //去抖延时时间
char    buf [256];                      //输入字符串缓冲

int config_card_for (int port,int address) {
        speed=9;
        combase=addressl
        if (port==SERIAL) {
                baudrate=12;            //选择38400波特
                timeout=7*speed *(baudrate+100);
                outportb(combase+3,131);      //准备PC串口通讯
                outportb(combase,baudrate);
                outportb(combase+1,baudrate/256);
                outportb (combase+3,3);
        }
        else {                                  /*PC-BUS将用来通讯*/
        baudrate=0;                             /*当波特率设为0，PC-BUS将用来通讯*/
        timeout=7*speed*100;
        outportb(combase+5,0);
        outportb(combase+6,0);
        }
}
int sendline (char *outchar) {                  /*在CR后呼叫发送字符用于给PMAC传送一
                                                  串字符*/
int i=0;
while (outchar[i]!=0)
        sendchar (outchar[i++]);
sendchar(13);
}
int sendchar (char outchar) {                  //给PMAC-PC卡传送一个字符
int i;
i=0;
if (baudrate==0) {                              /*由PC-BUS接口发数据*/
                                                /*查询写允许位（基地址+2的第1位）*/
        while (i++<timeout && !(inportb(combase+2)&2));
        if (i<timeout)
                outportb (combase+7,outchar);
}
else {                                          /*由串行RS232口发数据*/
                                                /*查询写允许位（基地址+5、6的第5、6位）*/
        while (i++<timeout && (inportb(combase+6)&16)==0);
        if(i<timeout)
                outportb(combase,outchar);
}
return(0);
}
int getline (char *linebuf) {
char ic;
```

```
int i,nc;
int i,nc;
if(able_to_talk) {
        ic=nc=i=0;
        if (baudrate==0) {                          /*由PC-BUS接口读数据*/
/*
                while (i++<timeout && ic!=13 && nc<255)
                        if ((inportb (combase+2)&1)==1) {
                                ic=inportb(combase+7);
                                linebuf[nc++]=ic;
                                i=0;
                        }
        }
        else {                                                  /*由RS-232接口读数据*/
                outportb(combase+4,2);          //取消延迟信号
                while (i++<timeout && ic!=13 && nc<255) {
                        if ((inportb (combase +5)&1)==1) {
                                ic=inportb(combase);
                                enable();
                                linebuf[nc++]=ic;
                                i=0;
                                disable();
                                }
                }
                outportb(combase+4,0);                  //设置延迟信号
                enable();
                while ((i+=2)<timeout);
        }
        linebuf[nc]=0;
        return(nc);                                     /*返回接收到的字符数*/
    }
    else{
        linebuf[0]=0;
         return(0);                                     /*没收到任何字符*/
    }
}

int Cards_On_Line() {                                   /*确认PMAC设备*/
int i,n;
able_to_talk=1;
n=i=0;
while (n<3 && i<11) {
        if(baudrate!=0) {                               /*串口通讯模式*/
           sendchar (24);                               /*发送 “^X”字符*/
          delay(20);
          sendchar(26);                                 /*发送 “^Z设置串行模式*/
        }
        sendline(“RHL:$720”);
        i=getline(buf);
        n++;
}
if (i>11 &&  i<17) {
        cprintf( “Found PMAC…\r\n”);
                while (getline (buf));                  /* 清除PMAC的输出缓冲 */
                return (1) ;
    }
        else {
                sound (1000) ;  delay (250 );  nosound ());
                cprintf ( “NO PMAC found ! \r\n) ;
```

```
                able_to_talk = 0 ;
                return (0 ) ;
        }
}

void main () {

        int done = 0,learning = 1;
        long position;
        char ic,prgram_line[80];

        config_card_for(PCBUS,528);
//      config_car_for(SERIAL,COM1);            //用此行代码进行PC-BUS通讯
        Cards_On_Line();
        if(able_to_talk) {
                cprintf("Now acting as a terminal to PMAC.\r\n");
                cprintf("Press <ESC> to abort.\r\n");
                cprintf("\r\nNow creating program 123.\r\n");
                cprintf("Press CTRL-B to learn current position of motor1.\r\n");
                cprintf("Press CTRL-D to see learned moves so far.\r\n");
                sendline("OPEN PROG 123 CLEAR CLOSE");
                while(!done){
                        while(!kbhit()) {               //无键按下，因此检查卡的数据
                                getline(buf);
                                cprintf(buf);
                        }
                        switch(ic = getch ()) {
                                case 2: //CTRL-B键按下，将‘X’传给程序
                                        sendline ( “#lp”);      //询问电机1的位置
                                        getline (buf);          //获取位置信息
                                        sscanf (buf, “%ld”,&position);
                                        sprintf ( program_line,”X%ld”,postion);
                                        sendline ( “OPEN PROG 123”);
                                        sendline (program_line);
                                        sendline (“CLOSE”);
                                        while (getline (buf));
                                        break;
                                case 4:                         //CTRL-D按下
                                        cprintf (“Learned moves in PROG 123:\r);
                                        sendline (“OPEN PROG 123 LIST CLOSE”);
                                        break;
                                case 27:                        //ESC键按下，退出
                                        done = 1;  break;
                                case 0:                         //取消特殊键的第二部分
                                        getch ();  break;
                                default:
                                        sendchar (ic);          //发给PMAC数据
                                        putch (ic);
                                        break;
                        }
                }
        }
        else
                cprintf (“Cant’t talk to PMAC:\r\n);
}
```

PMACRAM.C

/*PMAC中断驱动程序（PMAC选用方式2）
双端口RAM。请确认用于中断5的跳线E83已经装上以允许PMAC用IRQ5来中断PC，并且跳线E65已安装以允许EQU1在IRQ5上中断PMAC的8259。

```
/*--------------------------- PMAC bus port addresses -------------------------*/

#define PMACBUFFERSIZE   1024                  /*缓冲队列大小*/
#define PMAC             528                   /*PMAC总线地址*/
#define PMAC_STATUS      PMAC+2                /*总线状态（字节）*/
#define PMAC_DATA        PMAC+7                /*发送/接收的数据*/
#define PMAC_PIC00       PMAC+8                /*8259中断控制器*/
#define PMAC_PIC01       PMAC+9                /*8259中断控制器*/
#define PMAC_PIC02       PMAC+10               /*8259中断控制器*/
#define PMAC_RECEIVE     2                     /*接收允许位*/
#define PMAC_SEND        1                     /*发送允许位*/

#define PMAC_IS_SENDING (inp (PMAC_STATUS) & PMAC_SEND)

/*--------------------------- PMAC communication interrupt values----------*/

#define PMAC_IRQ         4                     /*0-7：IRQ0-IRQ7*/
#define PMAC_MASK        (~(1<<PMAC_IRQ))      /*取消PMAC 8259屏蔽*/
#define PMAC_EOI         0x20                  /*PMAC 结束中断*/
#define PMAC_IPOS        1                     /*IRQ0用于到位*/
#define PMAC_BREQ        2                     /*IRQ1用于缓冲要求*/
#define PMAC_ERROR       4                     /*IRQ2用于一般错误*/
#define PMAC_FERROR      8                     /* IRQ3用于跟随错误*/
#define PMAC_HREQ        16                    /* IRQ4用于通讯*/
#define PMAC_EQU1        32                    /* IRQ5用于双端口RAM标识*/
#define PMAC_NOP         0x40                  /*8259空操作码*/

/*----------------PC interrupt values---------------------------------------*/

#define IRQ              5                     /*PMAC占用的PC中断*/
#define PC_INT           (8+IRQ)
#define PC_MASK          (~(1<<IRQ))           /*PC 8259取消屏蔽*/
#define PC_PIC01         0x21                  /*8259中断控制器XT*/
#define PC_PIC00         0x20                  /* “0-7中断“*/
#define PC_EOI           0x20                  /*中断结束*/

#define ACK              0x06
#define LF               0x0C
#define EOL              0x0D
#define PCBUS            123
#define TRUE             1
#define FALSE            0

#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <stdlib.h>

static void      (interrupt far *old_comm_int) ();
static void      interrupt far pmac_comm ();

static int  combase, speed, able_to_talk, timeout, old_int_state, squl_flag;
static long far              *X, *Y, *Z;
static int  far              *M155, *M154;
static char                   buf [256];
```

```
void init_pmac_comm () {                                    /*初始化PMAC中断通信端口*/

        disable ();                                          /*关中断*/
        old_comm_int = getvect (PC_INT);                     /*保存旧的中断向量*/
        setvect (PC_INT, pmac_comm);                         /*写入新的中断向量！*/

/*-------------------设置PC的8259----------------------*/
        old_int_stat = inp (PC_PIC01);
        outp (PC_PIC01, ( old_int_stat & PC_MASK));

/*-------------------现在设置 PMAC的8259-------------*/
        outp (PMAC,    0);                                   /*去除PMAC的中断*/
        outp (PMAC_PIC00,    0x17);                          /*0x17为边缘,0x1F为电平触发
                                                             （ICW1）*/
        outp (PMAC_PIC01,    0x08);                          /*数据总线的向量*/
        outp (PMAC_PIC01,    0x03);                          /*设置8086模式*/
        outp (PMAC_PIC01,    0Xdf);                         /*mask out ferr, err,inpos, breq
                                                              (ICW2)*/
        outp (PMAC,    0X81);                                /*DSP读允许*/
        enable ();                                           /*开中断*/
}

void restore_pmac_comm ( ) { /*恢复旧的中断向量*/

        disable ( ) ;
        setvect ( PC_INT, old_comm_int ) ;                   /*代替旧的PC向量*/
        outp ( PC_PIC01, old_int_stat );                     /*恢复原有状态*/
        outp (PMAC_PIC01,    0Xff ) ;                        /*关闭PMAC的8259*/
        enable ( ) ;
}

static void interrupt far pmac_comm ( void ) {   /*PMAC中断服务程序*/

        int       ch ;
        char      isr ;

        disable ( ) ;                                        /*关中断*/
        outp ( PMAC_PIC02, PMAC_NOP ) ;                      /*第一个INTA/pulse边缘提高*/
        outp ( PMAC_PIC00, PMAC_NOP ) ;                      /*保持第一个INTA/pulse的边缘*/
        outp ( PMAC_PIC02, 0x0B ) ;                          /*准备读ISR寄存器/
        isr = inp (PMAC_PIC00 ) ;                            /*读ISR寄存器*/
        if ( isr & PMAC_EQU1 )                               /*如果有EQU1中断，设个标志*/
                equ1_flag = 1;
        outp ( PMAC_PIC00, PMAC_NOP ) ;                      /*保持第二个INTA/pulse
                                                              的边缘*/
        outp ( PC_PIC00, PC_EOI ) ;                          /*发结束中断给PC的8259*/
        enable () ;                                          /*开中断*/
}

void config_card_for ( int address ) {

        speed = 9 ;
        combase = address ;
        timeout = 7 * speed * 100 ;
        outportb ( combase + 5, 0 ) ;
        outportb ( combase + 6, 0 ) ;
}

void sendchar ( char outchar ) {                             /*给PMAC发个字符*/
```

```
                int     i= 0 ;

                while ( i++ < timeout  &&  !(inportb ( combase + 2 ) )  ;
                 if ( i < timeout )
                outportb (combase+7, outchar ) ;
}

void sendline (char *outchar ) {                        /*给PMAC送个带CR的字符串*/

        int i = 0;
        while ( outchar [ i ] != 0 )
                sendchar (outchar [ i++] );
        sendchar (13 ) ;
}

int getline ( char *linebuf ) {                         /*这是PMAC的一个无角的读取*/

        char    ic;
        int             i, nc ;
        if ( able_to_talk ) {
                ic = nc = i = 0;
                while ( i++ < timeout && != 13 && nc < 255 )
                        if (( inportb ( combase +2 ) & 1 ) ==1 ) {
                                ic = inportb ( combase +7 );
                                linebuf [nc++] = ic ;
                                i = 0;
                        }
                linebuf [ nc ] = 0 ;
                return (nc ) ;                          /*PMAC有信息要发*/
        }
        else {
                linebuf [0] = 0 ;
                return ( 0 ) ;                          /*PMAC无信息要发*/
        }
}

int Cards_On_line () {                                  /*检验PMAC 存在*/

        int     i,n ;

        able_to_talk = 1 ;
        n = i = 0 ;
        while ( n < 3 && i < 11 ) {
                sendline ( “RHL:$720” ) ;
                i = getline ( buf ) ;
                n++ ;
    }
    if ( i > 11 && i < 17 ) {
        cprintf ( “\r\nFound PMAC.....\r\n”);
        while (getline (buf ));                         /*清除PMAC的输出缓冲*/
        return ( 1 );
    }
        else {
                sound ( 1000 );  delay (250 ) ;  nosound ( ) ;
                cprintf ( “\r\nNo PMAC found! \r\n”) ;
                able_to_talk = 0;
                return ( 0 ) ;
        }
}
```

```
void send_command_to_PMAC () {                          /*将计算的位置给PMAC*/

        *x-= 100；                                      /*计算下一步的新位置*/
        *y-= 100；
        *z-= 100；
        *M155 = 1；                                     /*告知PMAC读位置*/
        cprintf ( “ Sending :  X%ld Y%ld Z%ld\r” , *X,*Y,*Z )；
}

void main () {

        int             done = FALSE；
        char    str [256]；

        config_card_for (528)；                         /*在PC-BUS之528处设置*/
        Cards_On_line ()；                              /*检验PMAC 存在*/
        Equl_flag = 0；
        if ( able_to_talk ) {
                sendline ( “WX:$786, $D,$43”)；         /*在PC的DPRAM的
                                                          $D4000处设置*/
                sendline ( “save”)；
                sendline ( “$$$”)；                     /*初始化卡以使DPRAM工作*/
                delay ( 1000)；                         /*确认I180,I280,I380等等以设为
                                                          1*/
                cprintf ( “Now sending positions via dual ported RAM &
interrupts ... \r\n\n”)；
                sendline (“Q &1 #1->X”)；               /*设置坐标系统*/
                sendline (“#3->Y”)；
                sendline ( “#4->Z”)；
                sendline ( “#1hm #3hm #4hm”)；          /*电机归原位*/
                sendline ( “i8=0”)；                    /*为每个伺服中断例程设置实时例程*/
                sendline (“M112->X:$C000,12,1”)；       /*EQU1输出使能为非*/
                sendline (“M113->X:$C000,13,1”)；       /*指向EQU1的输出反向使能*/
                sendline (“M116->X:$C000,16,1”)；       /*指向EQU1的比较相等标志*/
                sendline (“M151->DP:$D201”)；           /*用于电机X的指定位置*/
                sendline (“M152->DP:$D202”)；           /*用于电机Y的指定位置*/
                sendline (“M153->DP:$D203”)；           /*用于电机Z的指定位置*/
                sendline (“M154->y:$D200,0,1”)；        /*运动程序停止标志*/
                sendline (“M155->x:$D200,0,1”)；        /*PMAC读pos的握手标志*/
                getline (buf)；
                sendline (“OPEN PROG 10 CLEAR”)；
                sendline ( “TA2”)；                     /*设加速度时间为2毫秒*/
                sendline (“SPLINE1”)；                  /*设置做样条运动*/
                sendline (“X0Y0Z0”)；                   /*需要3个运动（2个提前作完）*/
                sendline (“X0Y0Z0”)；                   /*做3次样条拟和运动*/
                sendline (“WHILE(M154=1)”)；            /*循环直至被停止*/
                                                    sendline(“
                IF(M155=1)X(M151)Y(M152)Z(M153)M155=0M113=0M113=1”)；
                 sendline (“ENDWHILE”)；
                        /*M155由PC设为1，以靠知PMAC新的位置参数可以取了。M113先设为0再为
                 1的中断脉冲，以中断PC告知送新的位置参数。*/
                sendilne (“CLOSE”)；
                getline(buf):
                sendline (“M113=0”)；                   /*清除IR5的中断输入*/
                sendline (“M151=0”)；                   /*初始化电机X的位置*/
                sendline (“M152=0”)；                   /*初始化电机Y的位置*/
                sendline (“M153=0”)；                   /*初始化电机Z的位置*/
                sendline (“M154=1”)；                 /*运行运动程序*/
                sendline (“M155=1”)；                 /*立即读第一位置*/
```

```
            while ( getline(buf))；
            X=MK_FP(0xD400,0x0804)；          /*设置指针变量的指向*/
            Y=MK_FP(0xD400,0x0808)；          /*DPRAM的正确位置*/
            Z=MK_FP(0xD400,0x080C)；          /*X，Y，Z是位置变量*/
            M155=MK_FP(0xD400,0x0802)；       /*告诉PMAC计算数据的
                                                标志*/
            M154=MK_FP(0xD400,0x0800)；       /*停止PMAC程序的标志*/
            Init_pmac_comm ()；               /*设置事务处理中断向量*/
            sendline (“B10 R”)；              /*开始运行*/
            while(!done) {
          /*Now process the interrupt(s) */
                  if (equl_flag) {            /*如果有EQU1中断*/
                    equl_flag = 0；           /*清除标志*/
                    send_command_to_PMAC ( )；/*下载给PMAC计算的运动*/
                  }
        /* Background tasks can be inserted here!....  */
                  if (hbhit ())               /*检索键盘*/
                  if (getch () ==27)          /*按下ESC退出程序*/
                    done = TRUE；
            }
            *M154 = 0；                       /*停止运动程序*/
            restore_pmac_comm ()；            /*恢复原先中断设置*/
            sendline (“i8=2”)；               /*为每个3次样条中断设置实时中断处理*/
      }
}
```