# MTAction 运动控制脚本语言手册
# V1.07

## 1 版本历史

| 版本号 | 修改时间 | 修改内容 | 备注 |
|---|---|---|---|
| 1.00 | 2014-11-29 | 建立本文档 | |
| 1.01 | 2014-11-30 | 增加 UPDATE 指令<br>增加中文指令<br>修改部分指令名称<br>修改指针的实现方式，增加指针操作指令<br>增加程序控制指令 | 从文件中更新变量数据,实现重复动作 |
| 1.02 | 2014-12-20 | 细化指令说明和例程 | |
| 1.03 | 2014-12-21 | 增加插补急停指令和优化循环指令 | |
| 1.04 | 2014-12-27 | 增加内部时间计数读取指令,方便时间精度要求高的场合，精度 1ms | |
| 1.05 | 2015-02-12 | 修改逻辑判断一节的描述错误 | |
| 1.06 | 2015-03-13 | 修改加减乘除运算的语法格式<br>增加用户存储器读写指令<br>增加逻辑与 或 非指令<br>增加求余指令 | 支持用户存储数据到运动控制卡 |
| 1.07 | 2015-05-13 | 修改部分文字错误 | |

# 2 写约定

**格式:**助记符 空格或者逗号 参数 1 空格或者逗号 参数 2 空格或者逗号

**参数:**参数既可以是常量，也可以是变量，也可以是指针

**注释:**//后面的文字为注释

**分割:**每条指令占用一行,不得共行

**常量:**例如 100 -2143

**变量**:例如 #0 #1 #102 #402

**序号**:运动轴和 IO 的序号都是从 0 开始计算，比如第一轴为 0,第一个通道的输出为 0

**助记符描述:**助记符支持英文和中文，推荐使用英文，工具支持辅助插入助记符。一般的书写方式为 英文助记符(中文助记符),只需要一种即可。

**大小写:**英文默认推荐大写，小写也识别

**数值单位:**加减速度的单位为 Hz/s(步/秒$^2$)，速度的单位为 Hz(步/秒)

**执行流程:** 大部分指令执行后会立即执行下一条指令，少部分会等待，请注意指令说明，也就是说可以同时发送非常多的指令；如果有先后顺序的，请注意使用等待和条件判断

# 3 全局资源

PC 版本提供至少 65536 个全局变量(#0~#65535)用于应用程序和脚本

语言进行交互，也可以用作脚本语言的临时变量。

# 4 指令描述

## 4.1 任务相关

TASK_START　/任务开始

TASK_STOP　/任务停止

TASK_END　/任务结束

说明:

1) 各任务之间是并行执行，没有先后顺序的动作请分配到不同的任务中。

2) 其它有效的脚本指令都需要在任务开始和任务结束之间

3) 如果某个任务只需要执行一次，可以使用任务停止指令

4) 执行任务结束后会自动到任务开始处重新执行

例程：

TASK_START

……

TASK_STOP

……

TASK_END

## 4.2 归零模式

HOME_ACC_DEC(归零加减速) 轴序号　加速度　减速度 //归零参数

HOME(归零) 轴序号 归零速度 //归零动作

HOME_STOP 轴序号 //中止归零动作

说明：

1)归零速度可正可负，正为向正方向找零点，负为向负方向找零点，碰到对应的零点或者限位后停止，并置当前位置为 0

例程：

HOME_ACC_DEC 1 5000 5000 //第二轴加减速 5000,空格分隔

HOME_ACC_DEC ,2, 10000,10000 //第三轴加减速 10000,英文逗号分隔

HOME 1 -5000 //第二轴向负方向找零点

HOME 2 5000 //第三轴向正方向找零点

WAIT_STOP 1 //等待运动停止,找到零点或者负限位,

WAIT_STOP 2 //等待运动停止,找到零点或者正限位

WAIT_STOP 等待中不执行本任务中的后续指令，其它指令执行后立即执行后续指令

## 4.3 定速模式

V_ACC_DEC(定速加减速速度) 轴序号 加速度 减速度

V_REL(相对定速) 轴序号 相对速度

V_ABS(绝对定速) 轴序号 绝对速度

V_STOP 轴序号

说明：

1)速度可正可负，如果速度为在-50 和 50 之间则停止

例程：

V_ABS　0　1000　//第一轴 1000 定速运动

DELAY　10000　　//延时 10s(10000ms)

V_REL 0　-2000　　//反向定速 1000-2000=-1000

DELAY　10000

V_ABS 0 0　//停止　也可以用 V_STOP　0

## 4.4 定位模式

P_ACC_DEC_V(定位加减速速度)　轴序号　加速度　减速度　最大速度

P_REL(相对定位)　轴序号　相对步数

P_ABS(绝对定位)　轴序号　绝对位置

P_STOP　轴序号　//中止定位过程

说明：

1)绝对和相对的步数可正可负

例程：

P_ACC_DEC_V　3　4000 4000 10000 //设置参数

P_ABS 3 0 //移动到原点

WAIT_STOP 3//等待完成

P_ABS 3 10000 //走到 10000 处

WAIT_STOP 3 //等待完成

P_REL 3 -1000//后退 1000 个单位

WAIT_STOP 3//等待完成

## 4.5 多轴联动参数

L_ACC_DEC_V(联动加减速速度) 插补组序号 加速度 减速度 最大速度

L_AXIS(联动轴) 插补组序号 插补轴序号 插补轴序号 插补轴序号…

L_REL(相对联动) 插补组序号 相对目标 相对目标 相对目标 …

L_ABS(绝对联动) 插补组序号 绝对目标 绝对目标 绝对目标 …

L_STOP 插补组序号

说明

1)一般都会提供 2 组及以上的联动插补，也就是说可以同时进行多组的联动插补，联动插补同时启动同时停止

2)插补轴和插补目标要对应，比如 3 个轴插补，就要三个目标，否则可能会出现不可预知的错误

3)多个插补组之间不能占用同一个轴

4）插补的最大速度是指需要移动距离最大的轴的速度

例程:

```
L_ACC_DEC_V  0  5000 5000 1000  //插补参数
L_AXIS  0  0,2,3  //参与插补的轴
L_REL  0  1000,2000,3000 //同时相对移动目标
WAIT_L_STOP 0 //等待完成
L_AXIS  0  0,1,2  //更改插补的轴
L_ABS 0 0,0,0  //移动到原点
WAIT_L_STOP 0 //等待完成
```

## 4.6 圆弧插补

C_ACC_DEC_V (圆弧加减速速度) 圆弧插补组序号 加速度 减速度 速度

C_AXIS(圆弧轴) 圆弧插补序号 轴序号 0 轴序号 1

CW_REL (相对顺圆) 圆弧插补序号 相对目标 0,相对目标 1,半径

CW_ABS(绝对顺圆) 圆弧插补序号 绝对目标 0,相对目标 1,半径

CCW_REL(相对逆圆) 圆弧插补序号 相对目标 0,相对目标 1,半径

CCW_ABS(绝对逆圆) 圆弧插补序号 绝对目标 0,相对目标 1,半径

C_STOP 圆弧插补序号

1)一般都会提供 2 组及以上的圆弧插补，也就是说可以同时进行多组的插补

2)顺圆是指顺时针插补，逆圆是指逆时针插补,坐标为标准的右手坐标系

3)多个插补组之间不能占用同一个轴

4)插补的最大速度是指圆弧的线速度

5)目标和半径都可正可负，半径为正为劣弧（小），半径为负为优弧（大）

6)如果半径不合适不能组成圆，则有可能不动作

C_ACC_DEC_V 0 5000 5000 1000 //插补参数

C_AXIS 0 0,2 //参与插补的轴

CW_REL 0 1000,2000,80000 //相对当前位置以 80000 的半径圆弧到 1000,2000

WAIT_C_STOP 0 //等待完成

C_AXIS　0　0,1 //更改插补的轴

CCW_ABS 0 0,0,-10000　//圆弧插补到原点

WAIT_C_STOP 0 //等待完成

## 4.7　急停

HALT_ONE /单急停 轴序号

HALT_ALL /全急停

说明:

1)紧急停止会停止所有的动作，机械可能会有惯性，慎用

例程:

WAIT_IN 0 0　//等待紧急开关按下,紧急开关接在输入通道 0

HALT_ALL　//停止所有动作

## 4.8　等待运动状态

WAIT_STOP(等待停止) 轴序号

WAIT_NEG (等待负限位) 轴序号

WAIT_POS(等待正限位) 轴序号

WAIT_ZERO(等待零位) 轴序号

WAIT_L_STOP(等待联动停止) 联动组序号

WAIT_C_STOP(等待圆弧停止) 圆弧组序号

说明：

1)等待指令会停止指令所在任务的执行，直到满足条件，不影响其它任务

## 4.9 等待输入状态

WAIT_IN(等待输入) 输入通道序号 电平(0,1)

说明：

1)等待指令会停止指令所在任务的执行，直到满足条件，不影响其它任务

2)输入默认是高电平(1),有开关闭合是 0

## 4.10 读内部时间计数器

S_TICK(读时间) 变量号

说明:

1)本指令读内部的时间计数器的值到指定的变量里，变量如何使用，由用户决定

2)内部时间计数器的精度为 1ms

例程：

S_TICK,#100 //时间计数器进变量 100

## 4.11读输入状态

S_IN(读输入) 输入通道序号 变量号

说明:

1)本指令读指定的输入信号到指定的变量里，变量如何使用，由用户

决定

例程：

S_IN,0,#100　//输入通道 0 的状态进变量 100

IF_E #100,0 //如果为 0（开关闭合）,则开启继电器 0

OUT 0,1

IF_END

## 4.12读电机状态

S_RUN(读运行) 轴序号 变量号 //读轴的运行状态，1 运行，0 停止

S_NEG(读负限位) 轴序号 变量号//读轴的负限位，1 有效，0 无效

S_POS(读正限位) 轴序号 变量号//读轴的正限位，1 有效，0 无效

S_ZERO(读零位) 轴序号 变量号//读轴的零位，1 有效，0 无效

P_NOW(读位置) 轴序号 变量号//读轴的当前逻辑坐标位置

ENCODER_NOW(读位置) 轴序号 变量号//读轴的当前编码器位置

S_L_RUN(读联动运行) 轴序号 变量号//读联动插补的运行状态，1 运行，0 停止

S_C_RUN(读圆弧运行) 轴序号 变量号//读圆弧插补的运行状态，1 运行，0 停止

## 4.13 输出

OUT(光隔输出) 输出通道序号 电平(1,0) //1 通电，0 断电

## 4.14 延时

DELAY(延时) 毫秒 //1000ms=1s

说明:

1)本指令在满足延时时间前不会执行后续指令，不影响其它任务

## 4.15 循环

LOOP_START(循环开始) 循环次数

LOOP_END /循环结束

说明:

1)循环目前最多支持 3 层，可以嵌套,嵌套时必须保证内层和外层的对应

2)可以用 GOTO 跳出循环

例程:

```
LOOP_START 10
   …//其它指令
   LOOP_START   20
      …//其它指令
      LOOP_START   30
      …//其它指令
      LOOP_END
      …//其它指令
LOOP_END
```

…//其它指令

LOOP_END

共循环的次数为 10*20*30=6000 次

循环执行顺序和其它编程语言一样,循环层 2->循环层 1->循环层 0

## 4.16 标号和跳转

LABEL(标号) 名称 (任意字符)

GOTO(跳转) 名称

说明：

1)GOTO 可以向前跳，也可以向后跳，但是不能超过所在任务范围内

例程:

LABEL L1

…

IF_G #10 0 //如果变量 10 大于 0

…

GOTO L1

ELSE

…

GOTO L2

…

IF_END

…

LABEL L2

…

## 4.17 判断

IF_E 如果等于 v1 v2 //v1=v2?

IF_G 如果大于 v1 v2 //v1>v2?

IF_GE 如果大于等于 v1 v2 //v1>=v2?

IF_L 如果小于 v1 v2 //v1<v2?

IF_LE 如果小于等于 v1 v2//v1<=v2?

IF_NE 如果不等于 v1 v2//v1<>v2?

IF_ELSE 否则 //不满足条件的情况下

IF_END 如果结束 //必须有

说明:

1)v1 和 v2 为变量或者常量

2) IF_ELSE 可以不需要

3)IF_xx 必须和 IF_END 配合

4)变量和变量比较 变量和常数比较，常数和常数比较都可以

## 4.18 运算

ASSIGN 赋值 v1 v2 //v1=v2

ADD 加 v1 v2 v3//v1=v2+v3

SUB 减 v1 v2 v3//v1=v2-v3

MUL 乘 v1 v2 v3//v1=v2*v3

DIV 除 v1 v2 v3//v1=v2/v3 (v3 不得为 0)

MOD 求余 v1 v2 v3 // v1=v1 mod v3 (v3 不得为 0) 例 5 mod 2=1

说明:

1)v1 必须为变量，v2 v3 可为变量可为常量

## 4.19 更新数据

UPDATE (更新数据)

说明:

1）本指令会通知应用程序进行一次变量数据更新操作，更新的变量的意义由控制系统应用程序和脚本编辑程序共同定义。

2）控制应用程序是指由厂家提供的专用或者通用运动控制系统,一般用户不需要本功能

3）详细的接口定义请参看相关的文档

## 4.20 读指针数据

POINTER(读指针) v1 v2

说明:

1)v1 v2 都是变量

2)v2 中的数据作为变量的序号，再将对应的变量值赋值给 v1

例程:

当前变量的值

#100=625　#0=23　#1=100

POINTER #0 #1 执行后#0=625

## 4.21 程序控制

PLC_RUN(程序启动)

PLC_PAUSE(程序暂停)

PLC_STOP(程序停止)

1)程序控制指令只能在第一个任务中使用，第一个任务不受程序控制指令的影响，适合使用在紧急中断又想继续动作的场合

2)本指令会影响其它任务

3)第一个任务是指在编写时文本最上面的 TASK_START TASRT_END 所代表的任务

## 4.22 逻辑操作

AND(与)结果变量　常数/变量 常数/变量 常数/变量…

OR(或)结果变量　常数/变量 常数/变量 常数/变量…

NOT(非)结果变量　常数/变量

1)AND 和 OR 将后面最多 7 个变量或者常数进行逻辑与操作后，存入结果变量，供进一步使用

2)NOT 将后面的常量或者变量进行逻辑取反后存入结果变量

3)定义>0 为真 ，<=0 为假

### 4.23 用户存储

LOAD(装载变量)　变量　存储地址

SAVE(存储变量)　　变量　存储地址

例如:

LOAD #3 100 将第 100 个用户存储器地址的数据读入到变量 3

LOAD #3 #0　将变量 0 的数据做用户存储器的地址后，将数据读入变量 3

SAVE #3 100 将变量 3 的内如存储到地址为 100 的用户存储器中

1)读取和存储指令执行时间在 ms 级别，请尽量在启动时进行参数读取

2)存储指令会影响存储器寿命，一般为 10 万次，建议不要频繁读写，给一个信号后写入

3)存储器的内如可以通过通用工具在电脑上写入，某些场合用户只读即可

# 5 例程

### 5.1 流水灯

```
TASK_START
LABEL LED
OUT 1,0          //输出点 1  输出低
```

```
DELAY 1000      //延时 1000ms
OUT 1,1         //输出点 1 输出高
Delay 1000    //延时 1000ms
GOTO LED        //继续执行
TASK_END
```

## 5.2 归零后连续定长运动

```
TASK_START
HOME_ACC_DEC    0,2000,2000    //设置第一轴加减速
HOME 0,-3000    //负向找零点
WAIT_STOP 0      //等待停止
HOME 0,500    //以较慢的速度再找一次，提高 0 点精度
WAIT_STOP 0 //等待停止
P_ACC_DEC_V 0,2000,2000,2000 //定长运动的加减速和速度
LABEL CUT
P_REL 0,10000    //相对移动 10000 个单位
WAIT_STOP 0        //等待停止
OUT 0,1 //开启某个继电器
Delay 1000 //延时 1s
OUT 0,1 //关闭继电器
GOTO CUT //继续下一段
TASK_END
```

## 5.3 按键加减速

```
//参数设置任务
TASK_START
V_ACC_DEC 1 5000 5000      //设置好调速参数
TASK_STOP //本任务结束
TASK_END
//加速任务
TASK_START
LABEL L1
WAIT_IN 0 0//等待按钮 1  按下有效 ，默认为高，低电平有效
V_REL 1 1000    //提速
WAIT_IN 0 1 //等待按钮释放
DELAY 100 //避免重复判断动作
GOTO L1 //等待下一次动作
TASK_END
//减速任务
TASK_START
LABEL L2
WAIT_IN 1 0//等待按钮 1  按下有效 ，默认为高，低电平有效
V_REL 1 -1000    //提速
WAIT_IN 1 1 //等待按钮释放
DELAY 100 //避免重复判断动作
```

```
GOTO L2 //等待下一次动作
TASK_END
//停止任务
TASK_START
LABEL L3
WAIT_IN 2 0//等待按钮 1  按下有效 ，默认为高，低电平有效
V_STOP    1
WAIT_IN 2 1 //等待按钮释放
DELAY 100 //避免重复判断动作
GOTO L3 //等待下一次动作
TASK_END
```

### 5.4 JOG

```
//参数设置任务
TASK_START
V_ACC_DEC 1 5000 5000      //设置好调速参数
TASK_STOP //本任务结束
TASK_END
//正向点动任务
TASK_START
LABEL L1
WAIT_IN 0 0//等待按钮 1  按下有效 ，默认为高，低电平有效
```

```
V_REL 1 1000    //移动
DELAY 100 //避免重复判断动作
WAIT_IN 0 1 //等待按钮释放
V_STOP 0    //停止
GOTO L1 //等待下一次动作
TASK_END
//反向点动任务
TASK_START
LABEL L2
WAIT_IN 1 0//等待按钮 1  按下有效 ，默认为高，低电平有效
V_REL 1 -1000    //提速
DELAY 100 //避免重复判断动作
WAIT_IN 1 1 //等待按钮释放
V_STOP 1        //停止
GOTO L2 //等待下一次动作
TASK_END
```

### 5.5 交互直线切割

本例为说明使用进行了简化，假设进行的都是方形的切割块，切割的坐标由应用程序提供,Z 轴上下不考虑，初始移动点也不考虑，突出变量交互的使用，交互变量的定义如下:

#0 ： 启动标志，>0 开始，=0，等待启动

#1 ： >0 还有数据，=0 无有效数据，切割完成

#2 :X 坐标

#3: Y 坐标

```
TASK_START
L_ACC_DEC_V 0 5000 5000 5000
L_AXIS 0 0 1
LABEL FIRST
ASSIGN #0 0      //初始化
ASSIGN #1 0
ASSIGN #2 0
ASSIGN #3 0
IF_G #0 0    //if>0    启动
LABEL DO_CUT
   UPDATE    //更新数据
   IF_G #1 0 //有有效数据
     L_ABS 0 #2 #3    //2 轴联动插补
     WAIT_L_STOP 0    //等待插补完成
     GOTO DO_CUT       //读取下组坐标
   IF_ELSE       //本次切割完成
     GOTO FIRST
```

```
    IF_END
IF_END
GOTO FIRST
TASK_END
```