目 录

1.1 声明约定

特别注意!!! 本文的指令描述中：

- **COM** 表示指令名称；
- **X** 表示轴号或运动类型：

  0：X 轴；　1：Y 轴；　2：Z 轴；
  3：A 轴；　4：B 轴；　5：C 轴；
  6：XYZ 直线插补；　7：ABC 直线插补；
  8：XYZ 圆弧插补；　9：ABC 圆弧插补；
- **#data1,#data2、#data3** 等表示整数型常数；
- **#fdata1,#fdata2、#fdata3** 等表示整数型常数；
- **S#1,S#2** 表示整数型变量（或表示为寄存器），无特别说明，可以是 M 或 S 型变量,；
- **M#1,M#2** 表示为 M 型变量；
- **F#1,F#2** 表示为 F 型变量；
- **B#1,B#2** 表示位变量（或表示为位寄存器）
- **W#1,W#2** 表示为字变量；
- **#line** 表示为程序行号（整数型常数）
- 其它符号参照指令中的说明。

1.2 指令详解

1.2.1 运动类

1). 单轴运动

格式：　COM X,#data1 / COM X,S#1

X：指定轴号，有效值{0~5}（DMC630M 只能为 0~2）；
#data1、S#1：单轴运动数值，S#1 可以为 M 或 S 型变量；

指令：　DRVAD　绝对地址/常数值单轴运动
DRVAM　绝对地址/变量值单轴运动
DRVID　相对地址/常数值单轴运动
DRVIM　相对地址/变量值单轴运动

说明：　指令执行时，受控目标将沿单轴方向，移动#data（S#1）个脉冲数或移动到#data（S#1）位置（相对原点）。

例：Y 轴运动，当前坐标 Y=1000（与其他轴坐标无关），S130=3000。

指令："DRVAD 1,3000"　执行结果：系统从当前点，沿 Y 轴方向移动到 Y=3000 点，其他轴坐标不变，如图 1-1-A；

指令："DRVAM 1,S130"　执行结果：系统从当前点，沿 Y 轴方向移动到 Y=3000 点，其他轴坐标不变，如图 1-1-A；

指令："DRVID 1,3000"　执行结果：系统从当前点，沿 Y 轴方向移动到 Y=4000 点，其他轴坐标不变，如图 1-1-B；

指令："DRVIM 1,S130"　执行结果：系统从当前点，沿 Y 轴方向移动到 Y=4000 点，其他轴坐标不变，如图 1-1-B；

图 1- 1 单轴运动功能示意

2).直线运动

格式：　COM X,#data1,#data2,#data3 /　COM X,S#1,S#2,S#3

X：指定运动类型:，有效值{6,7}；

#data1,#data2,#data3/S#1,S#2,S#3:终点坐标或运动的数值;S#1,S#2,S#3 可以为 M 或 S 型变量;

指令： **LINAD** 绝对地址/常数值直线运动

LINAM 绝对地址/变量值直线运动

LINID 相对地址/常数值直线运动

LINIM 相对地址/变量值直线运动

说明： **1．坐标仅指插补指定轴的坐标；**

2．指令执行时,受控目标将沿从当前位置到目标点(#data1,#data2,#data3)或(S#1,S#2,S#3)的直线，按照设定速度运动。三个参数必须同为常数或者变量。

3．当执行运动指令时，如果所涉及的轴还在运动中，则等待该轴的运动停止后，执行下条指令，以下所有运动指令相同。

相对地址与绝对地址的判定：

相对地址： 以当前运动执行的起始点，得到的相对坐标数值;

绝对地址： 以坐标原点为基点的坐标数值。

使用相对地址和绝对地址的指令有区别：使用相对地址的指令其指令的倒数第二个字母为“Ｉ”，使用绝对地址的为“Ａ”;

常数值与变量值的判定：

常数值：以本身数值作参数； **变量值**：以变量中存储的数值作参数;

使用常数值或变量值的指令有区别：使用变量值的指令一般指令的最后一个字母为“M”。

注意：所有运动指令的地址坐标参数以控制器发出的脉冲数为单位。

例：当前 XYZ 坐标:(500,1000,800)，变量：S100=2000,S101=2500，S102=1500

指令：“LINAD 6,2000,2500” 执行结果：系统从当前点直线运动到点（2000,2500,1500），见图 1-10A;

指令：“LINID 6,2000,2500” 执行结果：系统从当前点直线运动到点（2500,3500,2300），见图 1-10B;

指令：“LINAM 6,S100,S101” 执行结果：系统从当前点直线运动到点（2000,2500,1500），见图 1-10A;

指令：“LINIM 6,S100,S101” 执行结果：系统从当前点直线运动到点（2500,3500,2300），见图 1-10B;

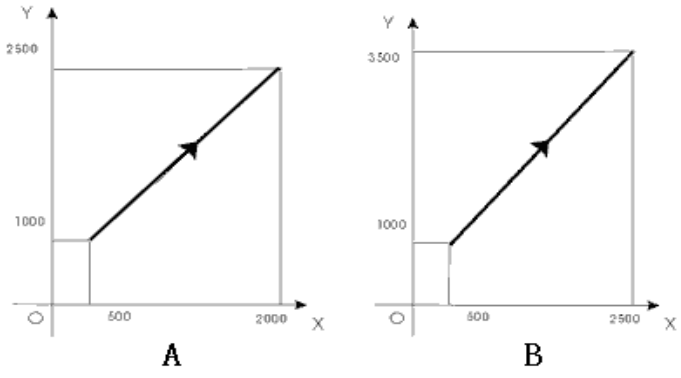

图 1- 10 直线插补运动功能示意

3).设定圆心（或圆弧中间点）

格式： **COM X,#data1,#data2,#data3 / COM X,S#1,S#2,S#3**

X：指定运动类型，有效值{8,9}；

#data1，#data2/S#1，S#2:圆心坐标或圆弧中间点坐标;

指令： **COID** 相对地址/常数值正向圆弧运动

COIM 相对地址/变量值正向圆弧运动

说明： **1.该指令需与圆弧插补指令配合使用，先设定圆心（或圆弧中间点），再由圆弧插补指令设定结束点坐标；**

2.具体为设定圆心或设定圆弧中间点，根据其后圆弧插补指令而定；

3.所有操作数必须同时为常数或者同时为变量；

4).圆弧插补

格式： COM X,#data1,#data2,#data3 / COM X,S#1,S#2,S#3

X：指定运动类型，有效值{8,9}；

#data1,#data2,#data3/S#1,S#2,S#3:圆弧插补终点坐标

指令： **CWAD** 绝对地址/常数值正向圆弧运动

CCWAD 绝对地址/常数值反向圆弧运动

CWAM 绝对地址/变量值正向圆弧运动

CCWAM 绝对地址/变量值反向圆弧运动

CWID 相对地址/常数值正向圆弧运动

CCWID 相对地址/常数值反向圆弧运动

CWIM 相对地址/常数值正向圆弧运动

CCWIM 相对地址/变量值反向圆弧运动

CWADP 绝对地址/常数值三点圆弧运动

CWAMP 绝对地址/变量值三点圆弧运动

CWIDP 相对地址/常数值三点圆弧运动

CWIMP 相对地址/变量值三点圆弧运动

说明： 1．运动类型为 8 时，实际为 XY 平面圆弧，Z 轴随动；运动类型为 9 时，实际为 AB 平面圆弧，C 轴随动；

2．正向圆弧运动：受控目标将沿以“已设定的圆心”为圆心，从当前点按照设定速度顺时针圆弧运行至终点坐标；

3．反向圆弧运动：受控目标将沿以“已设定的圆心”为圆心，从当前点按照设定速度逆时针圆弧运行至终点坐标；

4．三点圆弧运动：受控目标将从当前点按照设定速度经“已设定的圆弧中间点”圆弧运行至终点坐标；（不在同一直线的三个点可以确定一个唯一的圆弧路径）；

5．所有操作数必须同时为常数或者同时为变量；

6．须保证坐标值正确。如果终点不在以起点和圆心所确定的圆上，指令执行的结果有偏差，并不可预料。

5).运动停止

格式： COM X;

X：运动类型，有效值{0~9}；

指令： **STOP** 中止所指定的运动

说明：指令执行后，控制器将关断对应运动类型的脉冲。

例：指令：“STOP 2” 执行结果：关断 Z 轴单轴运动脉冲；

指令：“STOP 6” 执行结果：关断 XYZ 直线插补运动脉冲；

指令：“STOP 9” 执行结果：关断 ABC 圆弧插补运动脉冲；

1.2.2 命令控制类

1).速度设置

格式： COM X,#data1,#data2,#data3 / COM X,S#1,S#2,S#3

X：运动类型，有效值{0~9}；

#data1/S#1：起始速度；

#data2/S#2：加速度；

#data3/S#3：最高速度；

指令： **SPEED** 常数格式速度参数设置

SPEEDM 变量值格式速度参数设置

说明： 1．该指令对其后的运动有效；

2．速度曲线为标准梯形加速；

3．速度参数单位为脉冲频率，最低速度>1,最高速度<200000；

4．加速度越大，其速度曲线越陡峭；

5.对于插补速度的设置，其频率值对应于运动轨迹的线速度（即参与插补的两/三个轴的脉冲频率之和）；
6. 参数必须同为常数或者变量形式。

2).运动等待

格式： COM X
X:运动类型
指令： PAUSE 等待运动结束;
说明：1.指令执行后，系统直到当前指定运动类型的运动执行结束，再执行下一条指令。
2.可以使用判断指令（检测运动结束标记）实现运动等待功能；
例：指令："PAUSE 0" 执行结果:等待 X 轴的运动停止,然后执行下一条指令,如果 X 轴没有运动，则直接执行下一条程序;
指令："PAUSE 6" 执行结果:等待 X,Y,Z 三个轴或 XYZ 插补运动停止,然后执行下一条指令,,如果 X,Y,Z 轴都没有运动，则直接执行下一条程序;

3).延时

格式： COM #data / COM S#1
#data/S#1：延时时间,毫秒为单位;
指令： DELAY 固定时间延时
DELAYM 可变时间延时
说明：指令执行后，系统暂停 M1 毫秒的时间。"DELAY 0"可用作空指令(注:1 秒=1000 毫秒)；
例： S150＝3000;
指令："DELAY 200" 执行结果：延时 200 毫秒;
指令："DELAYM S150" 执行结果：延时 3 秒;

4).程序结束

格式： COM
指令：END 程序结束
说明：程序结束指令，用来标示用户程序结束，用户编程必需。
例：指令："END" 执行结果：当系统执行到该条指令后，程序停止执行，返回准待机状态。

5).速度改变

格式： COM X,#data1,#data2 / COM X,S#1,S#2
X：运动类型;
#data1，S#1：加速度;
#data2，S#2：最高速度;
指令：CHSPEED 常数格式速度改变指令
CHSPEEDM 变量格式速度改变指令
说明：该类指令适用于运动过程中的速度改变，例如遇到减速点。该指令执行后，所指定类型的运动速度将以改变至设定值。可以加速，也可以减速。

6).设定零点

格式： **COM**
指令：SETC 设置当前位置为零点（绝对坐标原点）
说明：该指令执行后，当前坐标值清零。
例：当前坐标 X=500,Y=600,Z=700,A=0,B=-1200,C=2500;
指令："SETC" 执行结果：X=0,Y=0,Z=0, A=0,B=0,C=0;

7).调用子程序

格式：**COM #line**

指令：CALL / 0216

说明：程序跳转至#line 行执行程序（自动保存当前程序行号），与"RET"指令配合使用；详见下一指令。

8).程序返回

格式： **COM**

指令： RET 子程序调用返回

RETI 中断程序返回

说明：1. RET 程序返回到"CALL"调用指令的下一行程序执行；

2.RETI 程序返回到中断程序执行前的指令的下一行程序执行；

例：

```
行号    指令
……
20      CALL 100            '跳转到"100"行程序执行
21      MOV M1,10
……
100     DRVID 0,2000
……
120     RET                 '返回到"21"行程序执行
```

9).程序调用

格式： **COM #data / COM #S1**

#data： 调用的程序序号；#S1： #S1 的值为程序序号；

指令： **CALLPROG** 常数格式调用程序,遇到 END 指令返回；

CALLPROGM 变量格式调用程序,遇到 END 指令返回；

说明：系统将执行指定的程序序号中的程序，遇到"END"指令返回至原程序执行位置的下一条指令；

10).运动结束提前量设置

格式： COM X,#data1 / COM X,S#1

X：运动类型，有效值{0~9}；

#data/#S1： 提前量数值，脉冲数为单位；

指令： **DRVFORE** 常数格式设置运动结束提前量

DRVFOREM 变量格式设置运动结束提前量

说明：1.指令执行后，再执行相应的运动指令，在运动结束前（提前量），系统自动设置标记-W301:B8(参见《CRT-DM600M 应用开发说明》-W301：系统运行状态 1)；

2.单轴运动直接为对应轴的脉冲数，插补运动为相关轴的脉冲数总数；

3．一般情况下，该指令需与判断指令配合使用；

1.2.3 跳转类

1).无条件跳转

格式： COM #line / COM M#1 / COM W#1

#line:跳转的目的行号；

S#1,W#1 寄存器指定的行号；

指令： **JMP** 跳转到指定行

JMPM 跳转到 M/S 型变量的值所指定的行

JWP 跳转到 W 变量的值所指定的行

说明： 1．执行该条指令，程序将跳转至#line 行或 M/S/W 变量指定的行接着执行。

2．可以跳转当前行的前面也可跳转到当前行的后面，但行号不能大于系统支持的最大行，不能小于 1（以下所有跳转指令相同）。

3．当程序中有出现多级的程序调用，即程序嵌套调用时，子程序返回建议使用"JMPM"；

例： 当前执行程序第 100 行，变量 M10＝11，

指令："JMP 10" 执行结果：程序跳转到第 10 行执行；

指令："JMPM M10" 执行结果：程序跳转到第 11 行执行；

2).位跳转

格式： **COM #line,B#1**

B#1:位变量;#line:目的行号;

指令： **JB** 指定位有效跳转；

JNB 指定位无效跳转；

JBM 以变量指定位，有效则跳转；

JNBM 以变量指定位，无效则跳转；

说明： 1．通用位变量(B0-B15)1 有效，0 无效；输入口位变量（I0-I7）0 有效，1 无效；

2．条件满足时跳转到指定行运行，否则执行下一条指令。

例：指令："JB 10,B2" 执行结果：如果 B2＝1，跳转到第 10 行程序；如果 B2＝0，接着执行下一条程序；

指令："JNB 10,B2" 执行结果：如果 B2＝0，跳转到第 10 行程序；如果 B2＝1，接着执行下一条程序；

指令："JB 15,I1" 执行结果：如果 I1 输入点有效（I1 为低电平），跳转到第 15 行程序；如果 I1 输入点无效（I1 为高电平），接着执行下一条程序；

指令："JNB 15,I1" 执行结果：如果 I1 输入点无效（I1 为高电平），跳转到第 15 行程序；如果 I1 输入点有效（I1 为低电平），接着执行下一条程序；

3).比较跳转

格式： COM #line,S#1,#data / COM #line,S#1,S#2

COM #line,W#1,#data / COM #line,W#1,W#2

#line 跳转的目的行号；

S#1,#data/S#1，S#2:相比较的两个数（可以为 M 或 S 变量）；

指令： **JGD** 变量值与常数值比较，大于则跳转；

JGM 变量值与变量值比较，大于则跳转；

JLD 变量值与常数值比较，小于则跳转；

JLM 变量值与变量值比较，小于则跳转；

JED 变量值与常数值比较，等于则跳转；

JEM 变量值与变量值比较，等于则跳转；

JNED 变量值与常数值比较，不等于则跳转；

JNEM 变量值与变量值比较，不等于则跳转；

JWGD 字判断：W 变量值与常数值比较， 大于则跳转；

JWGW 字判断：W 变量值与 W 变量值比较，大于则跳转；

JWLD 字判断：W 变量值与常数值比较， 小于则跳转；

JWLW 字判断：W 变量值与 W 变量值比较，小于则跳转；

JWED 字判断：W 变量值与常数值比较， 等于则跳转；

JWEW 字判断：W 变量值与 W 变量值比较，等于则跳转；

JNWED 字判断：W 变量值与常数值比较， 不等于则跳转；

JNWEW 字判断：W 变量值与 W 变量值比较， 不等于则跳转；

说明：当相应条件成立时跳转到指定行，否则执行下一条指令；

4).逻辑跳转

格式： **COM #line,W#1,#data / COM #line,W#1,W#2**

#line 跳转的目的行号；

W#1,#data/W#1, W#2:相比较的两个数；

指令： **JWAND** 字判断：W 变量值与常数值按位与,结果有效则跳转；

JWANDW 字判断：W 变量值与 W 变量值按位与,结果有效则跳转；

JNWAND 字判断：W 变量值与常数值按位与,结果无效则跳转；

JNWANDW 字判断：W 变量值与 W 变量值按位与,结果无效则跳转；

说明：当相应条件成立时跳转到指定行，否则执行下一条指令；

5).组合条件跳转

格式： **COM #line,S#1,S#2**

#line 跳转的目的行号；

S#1：指定 W 变量；S#2：指定判断条件；

指令： **JBX** 字判断：指定的 W 变量，满足条件则跳转；

JNBX 字判断：指定的 W 变量，不满足条件则跳转；

说明：1.当相应条件成立时跳转到指定行，否则执行下一条指令；

2.S#1,S#2 一般使用 M 变量；

3.W 变量指定，参见《CRT-DM600M 应用开发说明》- W 变量编址；

4.判断条件：32 位数值，从低位起，每 2 位一组，指定 W 变量的每 1 位判断条件：

01：有效；

10：无效；

00/11：不检测

6).循环跳转

格式： COM #line,#data / COM #line,S#1

#line：跳转的目的行号；

S#1：指定跳转执行次数；

指令： **LOOP** 循环跳转 N 次（N = **#Data**）；

LOOPM 循环跳转 S#1 次；

说明： 1．这两条指令用于建立循环，可以用常数指明次数，也可以用变量型指令以方便次数更改；

2．LOOP 指令不允许嵌套，即 Loop 指令与目的行之间不能再次出现该指令；如果必须使用嵌套循环，请使用 JMPM 指令实现。

例： M10＝100，

指令："LOOP 50,20" 执行结果：循环跳转到 50 行 20 次；

指令："LOOPM 30,M10" 执行结果：循环跳转到 30 行 100 次；

1.2.4 逻辑(位操作)

1).与

格式： **COM B#1, B#2**

指令： **AND** 逻辑与

说明： 双操作数都为位变量。B#1 位变量与 B#2 位变量"与",结果存于 B#1；当 B#1 为输出口位变量时，该指令影响输出口状态；B#1 不能为输入型 I/X。

例：当前 B1=1;B2=0, I1=0,O1=1,
　　指令："AND B1,B2"　　执行结果：B1=B1&B2=1&0=0,
　　指令："AND O1,I1"　　执行结果：O1=O1&I1=1&0=0,输出口 O1 有效;

2).或

格式：　**COM B#1, B#2;**
指令：　OR　　逻辑与
说明：　双操作数都为位变量。B#1 位变量和 B#2 位变量"或",结果存于 B#1;当 B#1 为输出口位变量时，该指令影响输出口状态；B#1 不能为输入型 I/X。
例：当前 B1=0;B2=1, I1=1,O1=0,
　　指令："OR B1,B2"　　执行结果：B1=B1|B2=0|1=1;
　　指令："OR O1,I1"　　执行结果：O1=O1|I1=0|1=1,输出口 O1 无效;

3).非

格式：　**COM B#1;**
指令：　CPL　　逻辑取反
说明：　操作数为位变量。该指令对 B#1 位变量的值取反;当 B#1 为输出口位变量时，该指令改变输出口状态；B#1 不能为输入型 I/X。
例：B1=0,O3=1;
　　指令："CPL B1"　　执行结果：B1=!B=!0=1;
　　指令："CPL O3"　　执行结果：O3=!O3=!1=0,输出口 O3 反转;

4).置位

格式：　**COM B#1;**
指令：　SETB　　置 1（有效）
说明：　操作数为位变量。该指令对 B#1 位变量的值置 1;当 B#1 为输出口位变量时，该指令使输出口有效（低电平）；B#1 不能为输入型 I/X。
例：指令："SETB B1"　　执行结果：B1=1;
　　指令："SETB O6"　　执行结果：O6=0,输出口 O6 有效;

5).清零

格式：　**COM B#1;**
指令：　CLRB　　置 0（无效）
说明：　操作数为位变量。该指令对 B#1 位变量的值清零;当 B#1 为输出口位变量时，该指令使输出口无效（高电平）；B#1 不能为输入型 I/X。
例：指令："CLRB B1"　　执行结果：B1=0
　　指令："CLRB O6"　　执行结果：O6=1,输出口 O6 无效;

6).按位与

格式：　COM W#1,#data1 / COM W#1,W#2;
指令：　WAND　　字操作：W 变量和#Data2 按位与（逻辑位操作）;
　　WANDW　　字操作：W 变量和#Data2 按位与（逻辑位操作）;
说明：　操作规则：参加运算的两个变量，若两者相应的位都为 1，则该位结果值为 1，否则为 0;

7).按位或

格式：　COM W#1,#data1 / COM W#1,W#2;
指令：　WOR　　字操作：W 变量和#Data2 按位或（逻辑位操作）;

WORW 字操作：W 变量和#Data2 按位或（逻辑位操作）；

说明： **操作规则：参加运算的两个变量，若两者相应的位有一个为 1，则该位结果值为 1，否则为 0；**

8).按位取反

格式： **COM W#1**

指令： **WCPL** 字操作：W 变量按位取反

说明： **W 变量的二进制数值，按位进行取反，即 0 变 1，1 变 0；**

9).移位

格式： **COM W#1,#Data / COM W#1, W#2**

指令： **WSFTL** 字操作：W 变量的值左移 N（#data）位

WSFTLW 字操作：W 变量的值左移 N（W#2 的值）位

WSFTR 字操作：W 变量的值右移 N（#data）位

WSFTRW 字操作：W 变量的值右移 N（W#2 的值）位

说明： **1.将 W 变量的二进制数值全部左移或右移若干位，移位后空白位补 0，而溢出的位舍弃；**

2.左移：高位移除，低位补零；右移：低位移除，高位补零；

1.2.5 运算类

运算指令的目的操作数（第一参数）必须为可读写变量（如 M/F 型变量）；
指令有两个参数的，结果存于第一参数中，第二参数值不变；

1).加

格式： **COM M#1,#data1 / COM M#1,S#2;**

COM F#1,#fdata1 / COM F#1,F#2;

COM W#1,#data1 / COM W#1,W#2;

指令： **ADD** 整型运算：M 变量加常数值

ADDM 整型运算：M 变量加变量值

FADD 浮点运算：F 变量加常数值

FADDM 浮点运算：F 变量加变量值

WADD 字运算：W 变量加变量值

WADDW 字运算：W 变量加 W 变量值

说明： **ADD 第二参数只能是整型数；第一参数如果是 F 变量，F 变量的小数部分将自动被删除；**

ADDM 第一参数如果是 F 变量，系统将取两个参数的整数部分相加，结果也只有整数部分；

FADD/FADDM 第一参数如果是 M 变量，系统将取两个参数的整数部分相加；

WADD/WADDW 结果溢出（大于 65535/0xFFFF）时，自动进位，结果只保留有效位（低 16 位）；

例：M1=200,M2=300,S100=400;

指令："ADD M1,500" 执行结果：M1=M1+500=200+500=700;

指令："ADDM M1,M2" 执行结果：M1=M1+M2=200+300=500;

指令："ADDM M1,S100" 执行结果：M1=M1+S100=200+400=600;

2).减

格式： **COM M#1,#data1 / COM M#1,S#2;**

COM F#1,#fdata1 / COM F#1,F#2;

COM W#1,#data1 / COM W#1,W#2;

指令： **SUB** 整型运算：变量减常数值

SUBM 整型运算：变量减变量值（M 或 S 型变量）

FSUB F 变量减常数值
FSUBM F 变量减变量值
WSUB 字运算：W 变量减变量值
WSUBW 字运算：W 变量减 W 变量值

说明： **SUB 第二参数只能是整型数；第一参数如果是 F 变量，其小数部分将自动被删除；**
SUBM 第一参数如果是 F 变量，系统将取两个参数的整数部分相减，结果也只有整数部分；
FSUB/FSUBM 第一参数如果是 M 变量，系统将取两个参数的整数部分相减；
WSUB/WSUBW 结果溢出（小于 0），只保留有效位（低 16 位）；

例： M1=200,M2=300,S100=400;
指令："SUB M1,500" 执行结果：M1=M1-500=200-500=-300;
指令："SUBM M1,M2" 执行结果：M1=M1-M2=200-300=-100;
指令："SUBM M1,S100" 执行结果：M1=M1-S100=200-400=-200;

3).乘

格式： **COM M#1,#data1 / COM M#1,S#2;**
COM F#1,#fdata1 / COM F#1,F#2;

指令： **MUL** 整型运算：变量乘以常数值
MULM 整型运算：变量乘以变量值
FMUL F 型变量乘以常数值
FMULM F 型变量乘以变量值

说明： **MUL 第二参数只能是整型数；第一参数如果是 F 变量，其小数部分将自动被删除（后运算）；**
MULM 第一参数如果是 F 变量，系统将取两个参数的整数部分相乘，结果也只有整数部分；
FMUL/FMULM 第一参数如果是 M 变量，系统将取两个参数的整数部分相乘；

例： M1=200,M2=12,S100=13;
指令："MUL M1,11" 执行结果：M1=M1*11=200*11=2200;
指令："MULM M1,M2" 执行结果：M1=M1*M2=200*12=2400;
指令："MULM M1,S100" 执行结果：M1=M1*S100=200*13=2600;

4).除

格式： **COM M#1,#data1 / COM M#1,S#2;**
COM F#1,#fdata1 / COM F#1,F#2;

指令： **DIV** 整型运算：变量除以常数值
DIVM 整型运算：变量除以变量值
FDIV F 型变量除以常数值
FDIVM F 型变量除以变量值

说明： **DIV 第二参数只能是整型数；第一参数如果是 F 变量，其小数部分将自动被删除（后运算）；**
DIVM 第一参数如果是 F 变量，系统将取两个参数的整数部分相除，结果也只有整数部分；
FDIV/FDIVM 第一参数如果是 M 变量，系统将取两个参数的整数部分相除；

例： M1=2000,M2=12,S100=13;
指令："DIV M1,10" 执行结果：M1=M1/10=2000/10=200;
指令："DIVM M1,M2" 执行结果：M1=M1/M2=2000/12=166;
指令："DIVM M1,S100" 执行结果：M1=M1/S100=2000/13=153;

5).清零

格式： **COM M#1 / COM F#1;**
指令： **CLR** 变量值清零;
说明：无;

6).自增

格式： **COM M#1 / COM F#1;**

指令： **INC** 变量 M#1 的值加 1;

说明： **1.M 型变量 M#1 每执行该指令一次，其值增一；多用于循环、计数。**
2.如果参数为 F 型变量，其值：整数部分加 1，小数部分自动清除；

例： 当前 M12＝10,
指令：“INC M12” 执行结果：M12=M12+1=11;

7). 自减

格式： **COM M#1;**

指令： **DEC** 整型运算：变量 M#1 的值减 1;

说明： **1. 如果参数为 F 型变量，每该指令执行一次，其值增一（多用于循环、计数）;**
2.如果参数为 F 型变量，其值：整数部分减 1，小数部分自动清除；

例： 当前 M12＝10,
指令：“DEC M12” 执行结果：M12=M12-1=9;

1.2.6 数据传送类

***浮点型 S/M 变量*：S/M 变量正常表示为整型变量；但触摸屏设备可以按 Modbus 地址将其定义为浮点 32 位数据，控制器读取时，可以使用指令“FMOVF”、“FLPF”获取正确的浮点数值，将其传送至 F 变量即可用于其他处理。**

1).赋值

格式： **COM M#1,#data2 / COM M#1,S#2 /**
COM F#1,#fdata2 / COM F#1,F#1 /
COM W#1,#data2 / COM W#1,W#1;

指令： MOV 整型操作：常数送入 M 变量;
MOVM 整型操作：S/M 变量的值送入 M 变量;
FMOV 浮点操作：常数送入 F 变量;
FMOVM 浮点操作：S/M 变量的值送入 F 变量;
FMOVF 浮点操作：***浮点型 S/M 变量***的浮点值送入 F 变量;
WMOV 字操作：常数送入 W 变量;
WMOVW 字操作：W#2 变量的值送入 W 变量;

说明： **MOV 第二参数只能是整型数；第一参数如果是 F 变量，F 变量的小数部分将自动被删除；**
MOVM 第一参数如果是 F 变量，系统将取两个参数的整数部分相加，结果也只有整数部分；
FMOV/FMOVM 第一参数如果是 M 变量，系统将取两个参数的整数部分相加；
FMOVF 第二参数为浮点数，第一参数一般为 S/M 型变量，

例： M0=101,S100=202,
指令：“MOV M2,100” 执行结果：M2＝100;
指令：“MOVM M2,M0” 执行结果：M2=M0=101;
指令：“MOVM M2,S100” 执行结果：M2=S100=202;

2).指针取值

格式： **COM M#1,S#2 / COM W#1,W#2**

指令： LP 整型操作：S#2 值指定的变量的整型值送入 M 变量
FLP 浮点操作：S#2 值指定的变量的浮点值送入 F 变量
FLPF 浮点操作：S#2 值指定的***浮点型 S/M 变量***的浮点值送入 F 变量

WLP　　字操作：W#2 值指定的变量的值送入 W 变量

说明：以变量 M#2 的值为"统一编码序号"的变量，将其数值送入 M#1 变量中；该指令可以认为是指针操作；

例：　M0=2,M3=1200,S200＝1890; M4＝3;

指令："LP M0,M3"　　执行结果：M0＝S200＝1890;

"LP M0, M4"　　执行结果：M0＝M3＝1200;

3).s 参数写入

格式：　**COM M#1,#data2　/　COM M#1,S#2**

指令：　SLD　　整型操作：数值存入 S 型变量；

SLM　　整型操作：变量数值存入 S 型变量；

SLP　　整型操作：#data2 指定的变量的数值存入 S 型变量；

SLPM　　整型操作：S#2 数值指定的变量的数值存入 S 型变量；

说明：1.S 变量赋值指令在程序中一定谨慎使用，　程序循环中频繁写入将造成控制器硬件损坏；

2．SLPM 的使用类似 LP；

1.3 指令编码表

（一） 运动类指令

指令名称	指令参数	说明
DRVAD	X,#data	指定某个轴运动到指定坐标，#data 为整型数，S#1 可以为 M/S
DRVAM	X,S#1	
DRVID	X,#data	指定某个轴运动若干数值，#data 为整型数，S#1 可以为 M/S
DRVIM	X,S#1	
LINAD	X,#data1,#data2,#data3	三轴直线插补命令，运动到目的坐标,参数必须同为常数或 M/S 变量
LINAM	X,S#1,S#2,S#3	
LINID	X,#data1,#data2,#data3	三轴直线插补命令，相对坐标参数,参数必须同为常数或 M/S 变量
LINIM	X,S#1,S#2,S#3	
COID	X,#data1,#data2,#data3	设定圆心/圆弧中间点，由其后圆弧插补指令确定具体功能
COIM	X,S#1,S#2,S#3	
CWAD	X,#data1,#data2,#data3	正向圆弧插补,并指定终点（绝对坐标），COID/COIM 指定圆心坐标
CWAM	X,S#1,S#2,S#3	
CCWAD	X,#data1,#data2,#data3	逆向圆弧插补,指定终点（绝对坐标），COID/COIM 指定圆心坐标
CCWAM	X,S#1,S#2,S#3	
CWID	X,#data1,#data2,#data3	正向圆弧插补，指定终点（相对坐标），COID/COIM 指定圆心坐标
CWIM	X,S#1,S#2,S#3	
CCWID	X,#data1,#data2,#data3	逆向圆弧插补，指定终点（相对坐标），COID/COIM 指定圆心坐标
CCWIM	X,S#1,S#2,S#3	
CWADP	X,#data1,#data2,#data3	三点圆弧，指定终点（绝对坐标），COID/COIM 指定圆弧中间点
CWAMP	X,S#1,S#2,S#3	
CWIDP	X,#data1,#data2,#data3	三点圆弧，指定终点（相对坐标），COID/COIM 指定圆弧中间点
CWIMP	X,S#1,S#2,S#3	
LLAD	X,#data1,#data2,#data3	指定三轴随前一运动（直线或圆弧插补）做三轴联动直线插补，当前
LLAM	X,S#1,S#2,S#3	以绝对坐标指定
LLID	X,#data1,#data2,#data3	指定三轴随前一运动（直线或圆弧插补）做三轴联动直线插补，当前
LLIM	X,S#1,S#2,S#3	以相对坐标指定
STOP	X	让某个轴/运动类型停止

（二） 命令控制类指令

指令名称	指令参数	说明
SPEED	X,#data1,#deta2,#deta3	设定某个轴/某类运动的起速,加速度,高速;
SPEEDM	X,S#1,S#2,S#3	参数必须同为常数或 M/S 变量
PAUSE	X	等待某个轴、某类运动的脉冲发完
DELAY	#data1	延时指令，延时多少时间，毫秒为单位
DELAYM	S#1	
END	---	程序结束
CHSPEED	X,#data1,#data2	使某个轴、某类运动的速度改变为设定值
CHSPEEDM	X,S#1,S#2	
SETC	---	设定原点，即坐标值清零

CALL	#line	调用某行的程序
RET	---	返回 CALL 指令调用的下一行
CALLPROG	#data1	调用子程序，遇 END 指令返回
CALLPROGM	S#1	
DRVFORE	X,#data	设置运动结束提前量
DRVFOREM	X,S#1	

（三） 跳转类指令

指令名称	指令参数	说明
JMP	#line	跳转到某行程序执行
JMPM	#line,#M1	同上，但行号存于 M 变量中
JWP	#line,W#1	同上，但行号存于 W 变量中
JB	#line,#B1	某个位变量有效的话，跳转到某一行
JNB	#line,#B1	某个位变量无效的话，跳转到某一行
JBM	#line,#M1	M 变量指定的位变量有效的话，跳转到某一行
JNBM	#line,#M1	M 变量指定的位变量无效的话，跳转到某一行
JGD	#line,S#1,#data1	如果指定的 M 或 S 变量大于某个数值的话，跳转到某一行
JGM	#line,S#1,S#2	指定的 M 或 S 变量大于另一个 M 或 S 变量，跳转到某一行
JLD	#line,S#1,#data1	如果指定的 M 或 S 变量小于某个数值的话，跳转到某一行
JLM	#line,S#1,S#2	指定的 M 或 S 变量小于另一个 M 或 S 变量，跳转到某一行
JED	#line,S#1,#data1	如果指定的 M 或 S 变量等于某个数值的话，跳转到某一行
JEM	#line,S#1,S#2	指定的 M 或 S 变量等于另一个 M 或 S 变量，跳转到某一行
JNED	#line,S#1,#data1	如果指定的 M 或 S 变量不等于某个数值的话，跳转到某一行
JNEM	#line,S#1,S#2	指定的 M 或 S 变量不等于另一个 M 或 S 变量，跳转到某一行
LOOP	#line,#data1	指定跳转到某一行若干次
LOOPM	#line,S#1	同上，但次数存于 S/M 变量中
JWGD	#line,W#1,#wdata	如果指定的 W 变量大于某个数值的话，跳转到某一行
JWGW	#line,W#1,W#2	指定的 W 变量大于另一个 W 变量，跳转到某一行
JWLD	#line,W#1,#wdata	如果指定的 W 变量小于某个数值的话，跳转到某一行
JWLW	#line,W#1,W#2	指定的 W 变量小于另一个 W 变量，跳转到某一行
JWED	#line,W#1,#wdata	如果指定的 W 变量等于某个数值的话，跳转到某一行
JWEW	#line,W#1,W#2	指定的 W 变量等于另一个 W，跳转到某一行
JWNED	#line,W#1,#wdata	如果指定的 W 变量不等于某个数值的话，跳转到某一行
JWNEW	#line,W#1,W#2	指定的 W 变量不等于另一个 W 变量，跳转到某一行
JWAND	#line,W#1,#wdata	如果 W 变量按位与某个数值，结果不为零（有效），跳转到某一行
JWANDW	#line,W#1,W#2	2 个 W 变量按位与，结果不为零（有效），跳转到某一行
JWNAND	#line,W#1,#wdata	如果 W 变量按位与某个数值，结果为零（无效），跳转到某一行
JWNANDW	#line,W#1,W#2	2 个 W 变量按位与，结果为零（无效），跳转到某一行
JBX	#line,S#1,S#2	S#1 指定的 W 变量，满足 S#2 指定的条件即跳转
JNBX	#line,S#1,S#2	S#1 指定的 W 变量，不满足 S#2 指定的条件即跳转

（四） 逻辑类指令（对位操作）

指令名称	指令参数	说明

AND	B#1,B#2	两个位变量进行与，存于第一个位变量
OR	B#1,B#2	两个位变量进行或，存于第一个位变量
CPL	B#1	将某个位变量取反
SETB	B#1	将某个位变量置位
CLRB	B#1	将某个位变量清零
WAND	W#1,#wdata	字操作：按位与，W#1 和 16 位数
WANDW	W#1,W#2	字操作：按位与，W#1 和 W#2
WOR	W#1,#wdata	字操作：按位或，W#1 和 16 位数
WORW	W#1,W#2	字操作：按位或 ,W#1 和 W#2
WCPL	W#1	字操作：W#1 按位取反
WSFTL	W#1,#wdata	字操作：W#1 左移#wdata 位
WSFTLW	W#1,W#2	字操作：左移 W#2 位
WSFTR	W#1,#wdata	字操作：右移#wdata 位
WSFTRW	W#1,W#2	字操作：右移 W#2 位

（五） 运算类指令

指令名称	指令代码	说明
ADD	M#1,#data	指定的 M 变量加上某一数值
ADDM	M#1,M#2	同上，但数值存于另一个 M 或 S 变量中
SUB	M#1,#data	指定的 M 变量减去某一数值
SUBM	M#1,M#2	同上，但数值存于另一个 M 或 S 变量中
INC	M#1	指定的 M 变量加 1
DEC	M#1	指定的 M 变量减 1
CLR	M#1	清指定的 M 变量
MUL	M#1,#data	指定的 M 变量乘于某一数值
MULM	M#1,M#2	同上，但数值存于另一个 M 或 S 变量中
DIV	M#1,#data	指定的 M 变量除以某一数值
DIVM	M#1,M#2	同上，但数值存于另一个 M 或 S 变量中
FADD	FM#1,#fdata	浮点加
FADDM	FM#1,FM#2	
FSUB	FM#1,#fdata	浮点减
FSUBM	FM#1,FM#2	
FMUL	FM#1,#fdata	浮点乘
FMULM	FM#1,FM#2	
FDIV	FM#1,#fdata	浮点除
FDIVM	FM#1,FM#2	
WADD	W#1,#wdata	字操作：加
WANDW	W#1,W#2	
WSUB	W#1,#wdata	字操作：减
WSUBW	W#1,W#2	

（六）数据传送类指令

指令名称	指令代码	说明
MOV	M#1,#data	数值赋给 M#1

MOVM	**M#1,S#2**	S#2 的数值赋给 M#1
FMOV	**F#1,#fdata**	数值赋给 F#1
FMOVM	**F#1,F#2**	F#2 的数值赋给 F#1
WMOV	**W#1,#wdata**	数值赋给 W#1
WMOVW	**W#1,W#2**	W#2 的数值赋给 W#1
FMOVF	**F#1,S#2**	***浮点型 S/M 变量***的值赋给 F#1
LP	**M#1,S#2**	将 S#2 变量中的值所代表的 S/M 变量的值赋给 M#1 变量
FLP	**M#1,S#2**	浮点取指
FLPF	**F#1,S#2**	将 S#2 变量中的值所代表的***浮点型 S/M 变量***的值赋给 F#1 变量
SLD	**S#1,#data**	S#1 变量写入:固定数值
SLM	**S#1,S#2**	S#1 变量写入:S#2 的数值
SLP	**S#1,#data**	S#1 变量写入: **#data** 所指定的参数的数值
SLPM	**S#1,S#2**	S#1 变量写入: S#2 所指定的参数的数值
WLP	**W#1,W#2**	W#2 指定的 W 变量值写入 W#1
WSLP	**W#1,w#2**	W#2 的值写入 W#1 指定的 W 变量