FANUC
数控宏程序编程案例手册

沈春根　汪健　刘义◎主　编

★ 数控车、数控铣和多轴加工宏程序相结合
★ 单型面、多型面和复合加工编程相结合
★ 编程思路、逻辑算法和刀路规划相结合
★ 程序内容、程序解释和编程提示相结合

微智造
机械人APP

QQ读者交流群：461463188

机械工业出版社
CHINA MACHINE PRESS

第1章 用户宏程序功能A

本章内容提要

FANUC数控系统提供两种用户宏程序功能：用户宏程序功能A（A类宏程序）和用户宏程序功能B（B类宏程序）。用户宏程序功能A是FANUC数控系统的标准配置功能。

本章首先详细介绍用户宏程序功能A的算术运算、逻辑运算、宏程序调用等基本知识，最后通过一个简单的实例详细叙述用户宏程序功能A在实际数控编程加工中的应用。

1.1 概述

用户宏程序功能A需要采用“G65 Hm”格式的宏指令来表达各种数学运算和逻辑运算，但是和用户宏程序功能B相比，不够直观，给阅读和编写都带来一定的难度，在实际生产中应用较少。但采用用户宏程序功能A来进行数控编程和加工产品，其思路、流程和算法等也适用于用户宏程序功能B和其他数控编程方式。

1.1.1 变量

FANUC数控系统中变量的定义是采用“#”和后面指定的变量号来表示，其中变量号可以是数值，也可以是表达式，甚至可以是其他的变量，如#100、#[3×2−1]等。

用变量可以代替地址后面的具体数字，当用变量代替变量号时，不能表示为“##101”或“#[#101]”，而应写成“#101”。

例如：#100 =10，G0 Z#100，相当于G0 Z10；

#101=10，G0 X[#101]，相当于G0 X10；

地址O和N后面不能引用变量，如O#100、N#100是错误的。

1.1.2 变量赋值

定义了变量号之后，数控系统会临时开辟一个内存字节来存放该变量，但该变量没有任何意义，只是向系统要一个空的内存，必须对该变量进行赋值后才能实现运算功能。

赋值格式：G65 H01 P#i Q#j;，其中H01为赋值运算符，相当于B类宏程序的赋值运算符“=”。

例如：G65 H01 P#200 Q1;，该语句的作用：把数值 1 赋值给#200 号变量。机床数控系统执行该语句后，#200 号变量就代表数值 1。当然变量的值在程序中也可以任意修改，如 G65 H01 P#200 Q3，就是将#200 修改为数值 3。

注意：P#200 与 Q1 位置不能互换。如 G65 H01 P#200 Q1，若误写成 G65 H01 Q1 P#200，则会触发系统报警。

Q 后面的数值为“-”，如 G65 H01 P#200 Q-1，该语句的作用：把数值-1 赋值给#200 号变量。

1.1.3 数学运算

（1）加法#i=#j+#k

格式：G65 H02 P#i Q#j R#k;，其中 H02 为加法运算符，P#i 存放的是 Q#j 与 R#k 相加的结果。

例如：G65 H02 P#201 Q#202 R#203;，该语句的作用：#201=#202+#203。

分析以下语句的意义：

G65 H01 P#200 Q1; ①

G65 H01 P#201 Q2; ②

G65 H02 P#202 Q#200 R#201; ③

① 的作用是将数值 1 赋值给#200 号变量，执行该语句后#200 号变量就代表数值 1，相当于#200=1。

② 的作用是将数值 2 赋值给#201 号变量，执行该语句后#201 号变量就代表数值 2，相当于#201=2。

③ 的作用是将#200 号变量的值加上#201 号变量的值，将两者相加的结果赋值给#202 变量，相当于#202=#200+#201。

（2）减法#i=#j-#k

格式：G65 H03 P#i Q#j R#k;，其中 H03 为减法运算符，P#i 存放的是 Q#j 与 R#k 相减的结果。

例如：G65 H03 P#201 Q#202 R#203 语句的作用：#201=#202-#203。

注意：G65 H03 P#i Q#j R#k 语句中 Q 后面的变量值是被减数，R 后的变量值是减数。

分析以下语句的意义：

G65 H01 P#200 Q1;

G65 H01 P#201 Q2;

G65 H03 P#202 Q#200 R#201;

请读者参考加法部分。

（3）乘法 #i=#j×#k

格式：G65 H04 P#i Q#j R#k;，其中 H04 为乘法运算符，P#i 存放的是 Q#j 与 R#k 相乘的结果。

例如：G65 H04 P#201 Q#202 R#203 语句的作用：#201=#202×#203。

（4）除法#i=#j/#k

格式：G65 H05 P#i Q#j R#k;，其中 H05 为除法运算符，P#i 存放的是 Q#j 与 R#k 相除

的结果。

例如：G65 H05 P#201 Q#202 R#203 语句的作用：#201=#202/#203。

注意：G65 H05 P#i Q#j R#k 语句中 Q 后面的变量值是被除数，R 后的变量值是除数。

（5）平方根 #i=$\sqrt{\#j}$

格式：G65 H21 P#i Q#j;，其中 H21 为平方根运算符，P#i 存放的是 Q#j 平方根的结果。

例如：G65 H01 P#202 Q9;　　④

G65 H21 P#201 Q#202;　　⑤

机床执行④、⑤后，#201 号变量值为 3，请读者分析。

（6）绝对值 #i=｜#j｜

格式：G65 H22 P#i Q#j;，其中 H22 为绝对值运算符，P#i 存放的是 Q#j 绝对值的结果。

例如：G65 H22 P#201 Q#202 语句的作用：#201=｜#202｜。

1.1.4　三角函数运算

（1）正弦函数#i=#j×SIN[#k]

格式：G65 H31 P#i Q#j R#k;，其中 H31 为正弦函数运算符，P#i 存放的是#j×SIN[#k]计算的结果。

例如：G65 H31 P#201 Q#202 R#203 语句的作用：#201=#202×SIN[#203]。

（2）余弦函数#i=#j×COS[#k]

格式：G65 H32 P#i Q#j R#k;，其中 H32 为余弦函数运算符，P#i 存放的是#j×COS[#k]计算的结果。

例如：G65 H32 P#201 Q#202 R#203 语句的作用：#201=#202×COS[#203]。

（3）正切函数#i=#j×TAN[#k]

格式：G65 H33 P#i Q#j R#k;，其中 H33 为正切函数运算符，P#i 存放的是#j×TAN[#k]计算的结果。

例如：G65 H33 P#201 Q#202 R#203 语句的作用：#201=#202×TAN[#203]。

1.1.5　逻辑运算

（1）逻辑或#i=#j OR #k

格式：G65 H11 P#i Q#j R#k;，其中 H11 为或运算符，P#i 存放的是 Q#j 与 R#k 进行 OR 运算的结果。Q#j 与 R#k 进行的是逻辑运算，逻辑运算的结果，其值只有 0 和 1（即 True 和 False）两种情况。如果运算的结果为 1，即条件表达式成立，否则条件表达式不成立。

例如：G65 H11 P#201 Q#202 R#203 语句的作用：#201=#202 OR #203。

（2）逻辑与#i=#j AND #k

格式：G65 H12 P#i Q#j R#k;，其中 H12 为与运算符，P#i 存放的是 Q#j 与 R#k 进行 AND 运算的结果。

例如：G65 H12 P#101 Q#102 R#103 语句的作用：#101=#102 AND #103。

逻辑运算运算规则：首先转换为二进制，然后按二进制运算规则进行运算。

本书以“AND”为例进行详细的叙述，其他逻辑运算符都可以按此进行分析。

例如：20 AND 1 进行以下步骤的运算：

第 1 步：将 20 转化为二进制数：20=（10100）。

第 2 步：10100 AND 1，其中 1 可以写出 00001，则按位运算 1 与 0 得 0、0 与 0 得 0、0 与 0 得 0、0 与 0 得 0、0 与 0 得 0。

第 3 步：20 AND 1 的值为 0，即 F。

1.1.6 跳转运算

（1）绝对跳转

格式：G65 H80 Pn；

意义：程序跳转到标号为 n 处（n 为跳转的标号），否则执行下一程序段，其流程框图如图 1-1 所示，示意图如图 1-2 所示。

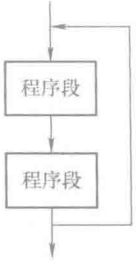

图 1-1 无条件转移语句流程框图

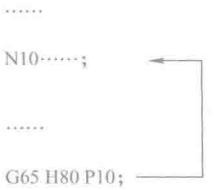

图 1-2 无条件转移语句示意图

例如：G65 H80 P10；无条件跳转到标号为 10 的程序处。

注意：使用该跳转语句时，必须要有跳转语句控制程序跳转到 G65 H80 P10 后面程序段处再执行后面的程序，否则会执行无限循环（死循环）。在程序设计中要正确使用该类语句。

（2）条件转移（EQ）

格式：G65 H81 Pn Q#j R#k；

意义：如果#j=#k，那么程序跳转到标号为 n 处（n 为跳转的标号），否则执行下一程序段，其流程框图如图 1-3 所示，示意图如图 1-4 所示。

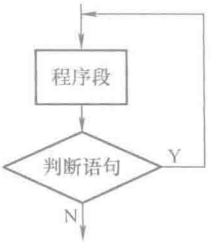

图 1-3 条件转移语句流程框图

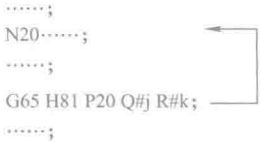

图 1-4 条件转移语句示意图

例如：G65 H81 P10 Q#101 R#102 语句的作用：当#101=#102，转移到 N10 程序段；若#101≠#102，执行下一程序段。

（3）条件转移（NE）

格式：G65 H82 Pn Q#j R#k；

意义：如果#j≠#k，那么程序跳转到标号为 n 处（n 为跳转的标号），否则执行下一程序段。

例如：G65 H82 P10 Q#101 R#102 语句的作用：当#101≠#102，转移到 N10 程序段；若#101=#102，执行下一程序段。

（4）条件转移（GT）

格式：G65 H83 Pn Q#j R#k；

意义：如果#j>#k，那么程序跳转到标号为 n 处（n 为跳转的标号），否则执行下一程序段。

例如：G65 H83 P10 Q#101 R#102 语句的作用：当#101>#102，转移到 N10 程序段；若#101≤#102，执行下一程序段。

（5）条件转移（LT）

格式：G65 H84 Pn Q#j R#k；

意义：如果#j<#k，那么程序跳转到标号为 n 处（n 为跳转的标号），否则执行下一程序段。

例如：G65 H84 P10 Q#101 R#102 语句的作用：当#101<#102，转移到 N10 程序段；若#101≥#102，执行下一程序段。

（6）条件转移（GE）

格式：G65 H85 Pn Q#j R#k；

意义：如果#j≥#k，那么程序跳转到标号为 n 处（n 为跳转的标号），否则执行下一程序段。

例如：G65 H85 P10 Q#101 R#102 语句的作用：当#101≥#102，转移到 N10 程序段；若#101<#102，执行下一程序段。

（7）条件转移（LE）

格式：G65 H86 Pn Q#j R#k；

意义：如果#j≤#k，那么程序跳转到标号为 n 处（n 为跳转的标号），否则执行下一程序段。

例如：G65 H86 P10 Q#101 R#102 语句的作用：当#101≤#102，转移到 N10 程序段；若#101 > #102，执行下一程序段。

1.1.7　调用

宏程序调用格式：G66 P<p>；

G67；

其中，p 为调用的子程序号；P 为调用的次数；G67 为取消宏程序调用指令。

宏程序调用示意图如图 1-5 所示。

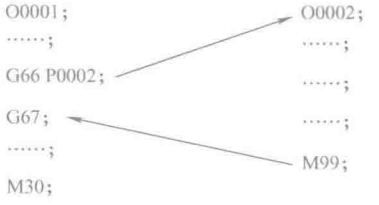

图 1-5　宏程序调用示意图

1.1.8　本节小结

本节介绍了用户宏程序功能 A 的基本概念和应用方法，它是 FANUC 数控系统的标准配置。采用 G65Hm 编程格式，给阅读和编写带来了一定的困难，因此在生产中很少使用。

编者在此只作简单叙述，生产中需要应用它的读者可以参考相关书籍。

1.2 简单应用

1.2.1 零件图以及加工内容

加工零件如图 1-6 所示，毛坯为ϕ50mm×100mm 圆钢棒料，需要加工成ϕ30mm×60mm 的光轴，材料为 45 钢，试采用用户宏程序功能 A 编写数控车加工宏程序代码。

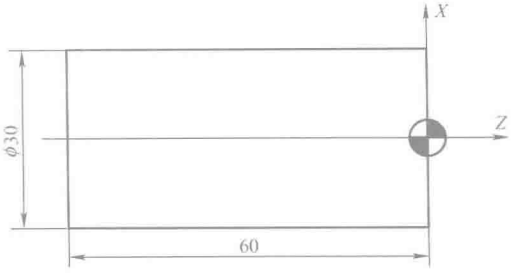

图 1-6 零件加工图

1.2.2 零件图样的分析

该实例要求车削成形一个ϕ30mm×60mm 光轴，在径向（X 轴）的直径余量为 20mm，加工和编程之前需要考虑以下方面：

1）机床：选择 FANUC 系统的数控车床。

2）装夹：普通自定心液压卡盘，夹持ϕ50mm×100mm 圆钢，伸出长度大约 65mm。

3）刀具：①90° 标准机夹式外圆车刀（1 号刀，粗加工）；②90° 标准机夹式外圆车刀（2 号刀，精加工）；③标准机夹式外圆切断刀（切槽刀），刀宽 3mm（3 号刀）。

4）量具：①0～150mm 游标卡尺；②25～50mm 外径千分尺；③0～150mm 深度游标卡尺。

5）编程原点：将编程原点放置在右侧端面中心，如图 1-6 所示。

6）车削余量 20mm（直径），车削外圆方式：分层车削；车削外圆模式：单向切削；背吃刀量：2mm。

7）切断外圆方式：分层、Z 轴正负方向进给切削方法，背吃刀量为 1mm。

8）设置转速和进给量见表 1-1。

表 1-1 车削光轴工序卡

工序	主要内容	设备	刀具（刀号）	切削用量		
				转速/（r/min）	进给量/（mm/r）	背吃刀量/ mm
1	车削端面	数控车床	外圆车刀（1 号刀）	2000	0.2	0.5
2	粗车外圆	数控车床	外圆车刀（1 号刀）	2000	0.2	2
3	精车外圆、倒角	数控车床	外圆车刀（2 号刀）	3000	0.12	0.3
4	切断外圆	数控车床	切槽刀（3 号刀）	350	0.03	1
5	车另一端面，倒角	数控车床	外圆车刀（1 号刀）	3000	0.12	0.3

1.2.3　算法以及程序流程框图设计

1.2.3.1　算法的设计

该实例是车削简单轴类零件，其加工要求和精度都不高，根据车削加工工序卡和数控编程的需要，对车削外圆ϕ30mm×60mm 的宏程序算法进行如下分析。

加工基本要求：ϕ30mm 和长度 60mm 为重要尺寸，加工时必须保证。该工序加工的基本思路：根据毛坯尺寸ϕ50mm×100mm，采用分层切削加工方式。

1）控制刀具快速移至 X50 Z0.5 后，*X* 轴每次进刀 2mm，*Z* 轴进行一次轴向行程的车削，车削长度为 63.3mm；*Z* 轴车削完成后，*X* 轴正方向增量退刀 0.5mm，*Z* 轴返回 Z0.5 处，此过程为一个循环周期，依此描述形成的单一循环刀路轨迹如图 1-7 所示。

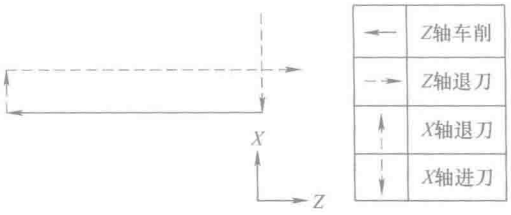

图 1-7　单一循环刀路轨迹

设置#100 号变量控制 *X* 轴尺寸，结合用户宏程序功能 A 赋值语句 G65 H01 P#200 Q50 来实现初始直径尺寸的赋值。

一个循环结束后，通过控制语句 G65 H03 P#200 Q#200 R2 实现 *X* 轴（径向）尺寸的变化。

2）选择 *X* 轴尺寸作为循环结束的依据。

通过循环语句 G65 H83 P70 Q#200 R30 实现整个车削过程的循环，多层的循环刀路轨迹如图 1-8 所示。

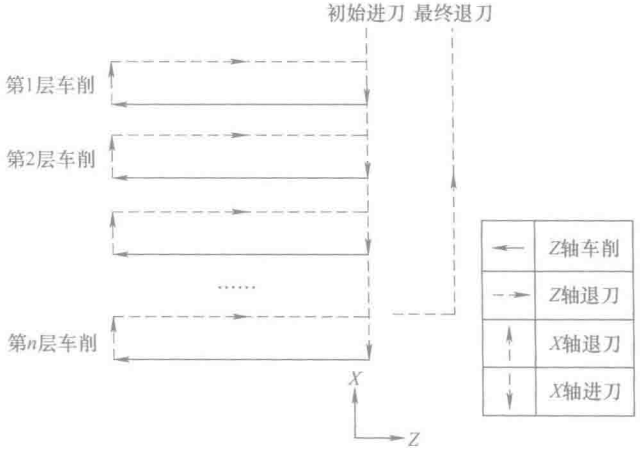

图 1-8　车削循环刀路轨迹

1.2.3.2　算法流程框图设计

根据以上算法设计和分析，规划程序流程框图如图 1-9 所示。

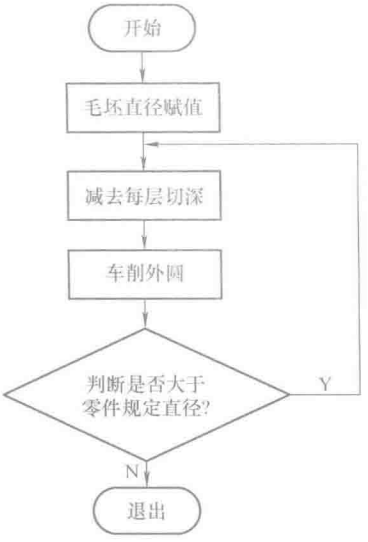

图 1-9　车削外圆流程框图

1.2.3.3　根据算法以及流程框图编写加工的宏程序代码

```
O0003;
N1;                          (车削端面，粗加工外圆)
T0101;                       (调用 1 号刀具及其补偿参数)
M03 S2000;                   (主轴正转，转速为 2000r/min)
G0 X50.5 Z50;                (X、Z 轴快速移至 X50.5 Z50)
Z0.5;                        (Z 轴快速移至 Z0.5)
M08;                         (打开切削液)
G01 Z0 F0.12;                (Z 轴进给至 Z0)
X-0.6;                       (X 轴进给至 X-0.6)
G0 Z1;                       (Z 轴快速移至 Z1)
X50.5;                       (X 轴快速移至 X50.5)
G65 H01 P#200 Q50;           (X 轴直径初始赋值 50mm)
N70 G65 H03 P#200 Q#200 R2;  (X 轴直径每次递减 2mm)
G65 H02 P#200 Q#200 R0.3;    (X 轴每次增加 0.3mm，0.3mm 为精加工余量)
G0 X#200;                    (X 轴快速移至 X#200)
G01 Z-63.3 F0.2;             (Z 轴车削长度 63.3mm)
G0 U0.5;                     (X 轴快速退刀，增量退刀 0.5mm)
Z1;                          (Z 轴快速移动至 Z1)
G65 H85 P70 Q#200 R30;       (条件判断语句，若#200 号变量的值大于 30mm，则跳转到
                              标号为 70 的程序段处执行，否则执行下一程序段)
G0 X100;                     (X 轴快速移至 X100)
Z100;                        (Z 轴快速移至 Z100)
M05;                         (主轴关闭)
M09;                         (关闭切削液)
N2;                          (精加工)
T0202;                       (调用 2 号刀具及其补偿参数)
M03 S3000;                   (主轴正转，转速为 3000r/min)
```

```
G0 X29 Z50;                (X、Z 轴快速移至 X29 Z50)
M08;                       (打开切削液)
Z1;                        (Z 轴快速移至 Z1)
G01 Z0;                    (Z 轴进给至 Z0)
X30 Z-0.5 F0.12;           (车削 C0.5mm 倒角)
Z-63.3;                    (车削 63.3mm 长度外圆)
G0 U0.5;                   (X 轴快速退刀，退刀增量为 0.5mm)
X100;                      (X 轴快速移至 X100)
Z100;                      (Z 轴快速移至 Z100)
M05;                       (关闭主轴)
M09;                       (关闭切削液)
M30;
```

编程要点提示

1）程序 O0003 是 FANUC 系统数控车床车削 ϕ30mm×60mm 粗加工、精加工用户功能 A 的加工代码。

2）程序 O0003 选择直径（*X* 向）作为变量，并作为循环结束判定依据。

3）车削 *Z* 向长度 63.3mm，是为了保证切断和车削另一端面的加工余量。

4）切断和车削另一端面、倒角宏程序代码不再详细给出，感兴趣的读者可以自行编写。

1.2.4　本节小结

1）本实例编程过程虽然简单，但体现了用户宏程序功能 A 编程的基本思路，可以作为学习宏程序功能 A 入门的实例。其中，设置合理的变量、变量之间的运算和选择判断语句（控制指令）是宏程序编程的关键。

2）用户宏程序功能 A 采用 G65 Hm P#i Q#i 格式编写宏程序代码，给阅读和编写带来一定的困难，在生产中应用较少，但其编程思路、变量赋值方法和逻辑算法等同样适用于用户宏程序功能 B 和其他编程方式。

本章小结

本章详细介绍 FANUC 系统用户宏程序功能 A 的变量赋值方式、数学运算、逻辑运算以及跳转等相关的基础知识，并通过一个简单车削加工实例详细介绍其编程的思路和加工工艺流程，给出了详细的用户宏程序功能 A 加工程序代码。

用户宏程序功能 A 由于自身的特点和局限性，在实际生产加工中应用较少。在用户宏程序功能 B 开发后，逐渐被淡忘。笔者只是进行了简单的介绍，感兴趣的读者可以参考相关的书籍，但其编程思路和算法等同样适用于用户宏程序功能 B 和其他编程方式。

第2章 用户宏程序功能B

本章内容提要

本章主要介绍用户宏程序功能B编程的基本知识，包括变量的定义、控制流向的语句（语法）等内容。其中，变量的定义是编写宏程序的基础，语法（即控制流向的语句）是编写宏程序的工具。最后通过讲解一个简单实例来介绍用户宏程序功能B编写程序的具体步骤和主要方法，为用户宏程序功能B编程以及其在数控加工上的应用打下基础。

2.1 编程基础——变量的定义

2.1.1 变量的概述

FANUC数控系统中，用户宏程序功能B变量的定义是用“#”和后面指定的变量号表示的，其中变量号可以是数值，也可以是表达式，甚至可以是其他的变量形式，如#100、#[3*2-1]等。

2.1.2 变量的赋值

定义了变量号之后，数控系统会临时开辟一个内存字节来存放该变量，但该变量没有任何意义，只是向系统要一个空的内存，必须对该变量进行赋值后才能实现运算功能。

赋值的格式为：变量=表达式。其中“=”为赋值运算符，其作用是把赋值运算符右边的表达式赋给左边的变量。

1）例如：#100=2是把数值2赋给变量#100，且变量#100是可以随时重新赋值的。

注意：赋值运算符两边的内容不能互换。如#100=#101+#102，如果误写成#101+#102=#100，意义就会截然不同：#100=#101+#102是#101+#102赋值给#100，而#101+#102=#100是#100赋值给#101+#102。

2）变量既可以参与运算，也可以相互进行赋值运算。

例如：
```
#100 = 1          把1赋给变量#100；
#101 = 2          把2赋给变量#101；
```

#103 = #100+#101　　把变量#100 的值加上变量#101 的值赋给变量#103；
#104 = #103　　把变量#103 的值赋给变量#104。

注意：在#104 = #103 这个赋值语句中，#103 必须有明确的值，如#103 没有明确的值，把#103 赋值给#104 是没有任何意义的。

2.1.3　变量的使用

1）变量号可以用变量代替。

例如：#[#100]号变量，设#100=101，则整个变量就等于变量#101。

2）#100 = 0 和#0 的区别。

#100 = 0 是把 0 赋值给变量#100，此时变量#100 就等于 0，有实际意义。

#0 为空变量，是永远不能被赋值的。

3）在地址后面指定变量号即可引用变量值。当使用表达式指定变量时，把表达式用“[ ]”括起来表明进行“[ ]”内部的运算；如果改变表达式符号时，要把符号放在表达式的前面。在编制宏程序变量时（而不是规定）最好用“[ ]”括起来，以免产生歧义。

例如：G01 X[2*#100+1] F#101；　　#101 是指定进给量；
G01 Z[-#100] F#101；　　#100 是地址符 Z 的数值。

4）变量用于条件转移和比较。

例如：IF [#100 GT #101] GOTO 20;，在条件转移语句中使用变量，增加了程序的灵活性。

5）在 FANUC 系统中，可以使用系统宏变量来设置坐标系、刀具长度数据、刀具半径等相关参数，比如：①#5221 表示 G54 第一轴（*X* 轴）零点偏移值；②#5222 表示 G54 第二轴（*Y* 轴）零点偏移值；③#5223 表示 G54 第三轴（*Z* 轴）零点偏移值；④#11001 表示 1 号刀具长度补偿量；⑤#10001 表示 1 号刀具长度磨耗补偿量；⑥#13001 表示 1 号刀具半径补偿量；⑦#12001 表示 1 号刀具半径磨耗补偿量。

6）有些场合不允许使用变量。

定义跳转的标号：GOTO#100，即使#100 有明确的赋值也不行。

定义程序名：O#100。

7）在实际编程中，每个变量要单独写一行，不能把多个变量写在同一行，否则系统会出现报警。

例如：

正确写法	错误写法
……	……
#100 = 10;	#100 = 10; #101 = 2; #102 = 0;
#101 = 2;	……
#102 = 0;	
……	

2.1.4　变量的类型

变量根据变量号中数字的范围，可以分为以下四种类型，具体见表 2-1。

表 2-1　变量的类型及功能

变量号	变量类型	功能
#0	空变量	该变量总是空的，没有值赋给该变量
#1～#33	局部变量	局部变量只能用在宏程序中存储数据，例如运算断电时，局部变量被初始化为空，调用宏程序时，自变量对局部变量进行赋值
#100～#199	公共变量	在不同的宏程序中意义相同，当断电时，变量#100～#199 初始化为空
#500～#999	公共变量	变量#500～#999 用于数据保存，即使断电也不丢失
#1000 以上	系统变量	系统变量用于读和写 CNC 的各种数据，例如刀具的当前位置和补偿值等

在实际编程中，可以使用变量号在公共变量范围中的变量（即#100～#999），这些变量是厂家提供给用户自由使用的；系统变量号中的变量出于厂家对系统的保护，是不可以随便写入数据改变其值的。关于 FANUC 0i 更多的系统变量、接口系统变量、采用系统变量读入和改写刀具补偿值、改写工件零点的偏置等的说明，读者可以参考 FANUC 公司提供的 FANUC 0i 数控系统操作使用说明书。

2.1.5　变量的算术运算和逻辑运算

表 2-2 所列为变量的算术运算和逻辑运算，应用时应注意以下几点：

表 2-2　变量的算术运算和逻辑运算一览表

功能	格式	备注
定义置换	#i=#j	变量的赋值
加法 减法 乘法 除法	#i=#i+#k #i=#i-#k #i=#i*#k #i=#i/#k	变量的数学运算
正弦	#i=SIN[#j]	三角函数和反三角函数的数值均以度为单位来指定。例如：90° 30′ 应写成 90.5°
反正弦	#i=ASIN[#j]	
余弦	#i=COS[#j]	
反余弦	#i=ACOS[#j]	
正切	#i=TAN [#j]	
反正切	#i=ATAN[#j]	
平方根	#i=SQRT[#j]	按四舍五入取整进行运算的
绝对值	#i=ABS[#j]	
舍入	#i=ROUND[#j]	
指数函数	#i=EXP[#j]	
自然对数	#i=LN[#j]	
上取整	#i=FIX[#j]	
下取整	#i=FUP[#j]	
与	#I AND #j	逻辑运算是按位进行运算的，按照二进制数据运算
或	#I OR #j	
异或	#I XOR #j	
从 BCD 转为 BIN	#I=BIN #j	用于与 PMC 信号交换（BIN 为二进制；BCD 为十进制）
从 BIN 转为 BCD	#I=BCD #j	

1）以上只是罗列了变量算术运算和逻辑运算的基本功能。在实际宏程序编制中，不是每个功能及其格式都需要用到，建议记住常用的一些命令，如定义置换、加减乘除、SQRT[ ]、SIN[ ]、COS[ ]等，其他的指令可以在实际编程中按需查询。

2）注意运算的优先次序。

函数→乘除（*、/、AND）→加减运算（+、−、OR、XOR）→赋值运算（=），该类运算是按从高到低的顺序执行的，其中赋值运算（=）优先级最低。

3）方括号的嵌套。

方括号“[ ]”用于改变运算的次序，方括号最多可以嵌套五级，包括函数内部使用的括号；圆括号“( )”则用于注释程序的含义。

例如：#100=SQRT[1−[#101*#101]/[#103*#103]];（两重括号）

4）关于上取整函数 FIX 和下取整函数 FUP，在实际应用时要注意使用后数值的变化。应用 FIX 函数时绝对值比原来的绝对值大，反之为下取整（建议在实际编程中尽量避免使用这些函数，以免产生零件精度和尺寸的误差）。

5）有关由于函数计算的误差而引起零件精度的问题，可以参考 FANUC 公司提供的 FANUC 0i 数控系统操作使用说明书，在此不再赘述。

2.2　编程工具——控制流向的语句

FANUC 系统提供的跳转语句和循环语句在程序设计者和数控系统之间搭建了沟通的桥梁，使宏程序编程得以实现。其中，跳转语句可以改变程序的流向，使用得当可以让程序变得简洁易读，反之，则会使程序变得杂乱无章。

2.2.1　控制语句的分类

1. *无条件跳转指令（也称绝对跳转指令）*

格式：GOTO n;（n 为标号，n 的范围为 1～99999，不在这个范围内，系统会自动报警，报警号 No.128）

语义：跳转到标号为 n 的程序段。

例如：

```
……;
N20……;  ←
……;
GOTO 20;
……;
```

注意：

1）GOTO n，其标号不可以使用变量。

例如：
```
#100 = 10;
GOTO #100;
```
机床执行上述两段程序后触发系统报警。

2）使用该跳转语句时，必须要有跳转语句使程序跳转到 GOTO n 后面程序段处，并执行后面的程序，否则会执行无限循环（死循环），如图 2-1 所示。

3）GOTO n 语句的使用场合。

① 一般用于实现程序的跳转（跳行），相当于 FANUC 系统控制面板上的“/”（选择

跳过）功能。

② GOTO n 语句和 IF [条件表达式] GOTO n（[ ]中为条件转移语句）配合使用，可实现编程者预期目的，如图 2-2 所示。

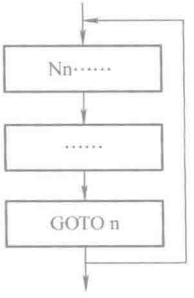

图 2-1 GOTO 语句跳转流程示意图

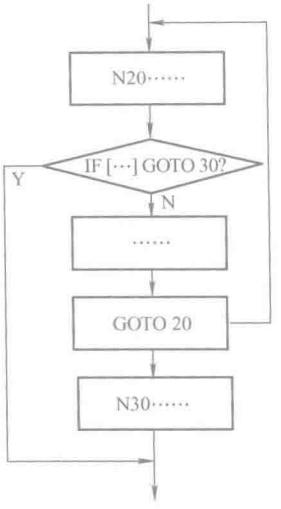

图 2-2 GOTO 语句用法跳转流程示意图

2. 条件转移语句

格式：IF [条件表达式] GOTO n;（n 为程序的标号）

语义：指定表达式满足时，转移到顺序号为 n 的程序段执行；如果指定的条件表达式不满足，则执行下一个程序段。

例如：

```
……;
N20 ……;
……;
IF [条件表达式] GOTO 20;
```

如果[ ]中表达式成立，则跳转到程序号为 20 处执行，否则执行下一个程序段，其流程框图如图 2-3 所示。

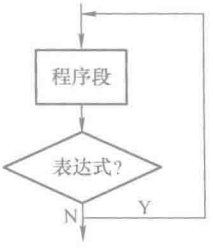

图 2-3 条件转移语句的流程框图

注意：在 FANUC 系统输入时，IF 和[条件表达式]之间必须有空格隔开，[条件表达式]和 GOTO 20 之间必须有空格隔开。条件表达式必须包含运算符（见后面详细说明）。运算符插在两个变量中间或变量和常量之间，必须加[ ]符号。

3. 条件赋值语句

格式：IF [条件表达式]　THEN　……；

语义：条件表达式满足时，执行 THEN 后面的语句；如果不满足，则执行 IF 程序段的下一条语句，该语句相当于对变量有条件赋值。

例如：　　……；

IF [条件表达式]　THEN　#100 = 0；

……；

如果[　]中表达式的条件成立，则执行#100=0 的语句；否则执行下一个程序段，该语句相当于内嵌 IF [　] GOTO n 语句，如图 2-4 所示。

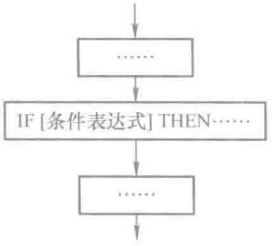

图 2-4　IF [ ] THEN ……语句执行流程

IF 语句可以嵌套 IF 语句，如下所示：

```
……;
N10 ……;
N20 ……;
……;
IF [条件表达式] GOTO 20;
……;
……;
IF [条件表达式] GOTO 10;
……;
……;
```

IF 语句也可以互相交叉的，如下所示：

```
……;
N20 ……;
N10 ……;
……;
IF [条件表达式] GOTO 20;
IF [条件表达式] GOTO 10;
……;
```

4. 循环语句（WHILE 语句）

格式：WHILE [条件表达式]　DO m；

循环体；

END m；　　（m 为程序段的标号）

语义：在 WHILE 后面指定了一个条件表达式，当条件表达式的值为 True（真）时，则执行 WHILE 到 END 之间的循环体的程序段；当条件表达式的值为 False（假）时，执行 END 后面的程序段，其程序执行的流程框图如图 2-5 所示。

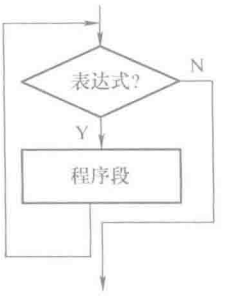

图 2-5　循环语句流程框图

关于 WHILE 语句的几点说明如下：

1）DO m 和 END m 必须成对使用，而且 DO m 必须在 END m 之前使用，是用识别号 m 来寻找和 DO 相配对的 END 语句，下面是错误的用法：

WHILE [条件表达式] DO 1；

循环体；

END 2；

2）其中 m 的取值只能为 1、2、3，如果使用 1、2、3 以外的数字，系统会报警，报警号为 No.126。

3）[ ]中的语句为条件表达式，循环的次数根据条件表达式来决定，如果条件表达式的值永远为 True，则会无限次执行循环体，即出现无限循环的现象。在进行程序设计时，要先设计好算法，避免出现无限死循环的现象。

例如：WHILE [1 GT 0] DO 1；

循环体；

END 1；

因为 1 永远大于 0，所以此语句会无限次地执行循环体中的程序段。

4）条件转移语句（IF [条件表达式] GOTO n）和循环语句（WHILE）的区别：两者的区别在于搜索方式不同，本质上没有太大区别，但在实际应用中要注意它们微小的区别。一般能用循环语句（WHILE）的语句都可以用 IF [条件表达式] GOTO n 来替代。

例如：

```
……；                                  ……；
#100 = 100；                           #100 = 100；
N20 #100 = #100-10；                   WHILE [#100 GT 20 ] DO 1；
……  ；                               #100 = #100-10；
程序段；                               程序段；
IF [ #100 GT 20 ] GOTO 20；            END 1；
……；                                  ……；
```

这两个程序的运行结果完全一样。再看下面的程序：

```
……；                                  ……；
#100 = 20；                            #100 = 20；
N20 #100 = #100-10；                   WHILE [#100 GT 20 ] DO 1；
……；                                  #100 = #100-10；
程序段 ；                              程序段；
IF [ #100 GT 20   ] GOTO 20；          END 1；
```