TORNOS

B Macro

ISO 机床

目录

1 让您的程序更智能！

1.1 “B Macro”是什么？

B Macro 是一种编程语言，其参数由 FANUC 数控装置设置。
全部最新的 ISO 机床都免费内置了这种语言。
它让您的程序更智能。我们将在第 3 章提供具体的使用范例。
最重要的是不要被表面迷惑，首次阅读时会感觉很复杂，但实际日常程序的使用极为简单。

1.2 您知道吗？

B Macro 是一种非常简单的编程语言，它为您提供了大量可能性。
例如，大部分 ISO 机床里 Tornos 宏都由两万多行这种语言的代码组成。

2 B Macro 基本说明

2.1 变量

为让程序智能，您要能将数值替换为变量。
变量是可赋值的信息段。
变量语法可通过“#”号及其识别码识别。

变量示例：#153
为变量赋值的示例： #153 = 12.4（这里，变量 #153 赋值 12.4）
变量赋值后，就可以进行加乘运算、可以用作位置或速度信息、作为馈送，甚至用作条件表达式。

以下是您可以使用的变量。

Null 变量：
Null 变量是从不赋值的变量。

变量是：#0

局部变量：
局部变量是在您退出已赋值程序后被初始化为“零”的变量。

这些变量是：#1 - #33

每个通道的全局变量：
每个通道的全局变量是在退出变量已赋值的程序时不会重置为“零”的变量。
事实上，这些变量按通道意味着，为变量赋值时，该值仅在为变量赋值的通道里。

这些变量是：#150 - #199

通用全局变量：
通用全局变量是在退出变量已赋值的程序时不会重置为“零”的变量。
事实上，称为通用变量意味着，为变量赋值时，这个变量将机床的所有通道中包含此值。

这些变量是：#600 - #699

系统变量：
系统变量可用于读写数控数据，比如对刀变量、轴位置数据等。
欲了解更多系统变量信息，请参阅 FANUC 说明。

这些变量是：> #1000

2.2 算术和逻辑运算

算术运算

最常用的算术运算是：

加	"+"
减	"-"
乘	"*"
除	"/"

使用示例： #603 = [#601 + #602] / 4

功能：

最常用的功能是：

正弦	"SIN[#...]"
余弦	"COS[#...]"
正切	"TAN[#...]"
平方根	"SQRT[#...]"
绝对值	"ABS[#...]"
功率	"POW[#..., #...]"
向下取整到下一个数	"FIX[#...]"
圆整值	"ROUND[#...]"

使用示例： #603 = COS[#602]

注意：角度单位是度。例如：90 度和 30 分写做 90.5 度。

关系操作符：

关系操作符可用于条件表达式的两个变量比较。

关系操作符是：

等于	"EQ"
不等于	"NE"
大于	"GT"
大于等于	"GE"
小于	"LT"
小于等于	"LE"

逻辑运算：

逻辑运算可以在一个条件表达式中测试多个条件。

最常用的逻辑运算是：

逻辑与	"AND"
逻辑或	"OR"

2.3 连接和重复

"GOTO"条件说明：

位于行开始位置的指令"Nn"可表示块编号。
指令"GOTOn"很容易理解，GOTO 5 表示：直接跳到块 N5。

使用示例：

```
...
GOTO 5
G0 X2 T35
M103 S200
G0 Z6
N5 G0 X3 T26
...
```

数控不运行这部分程序

条件指令"IF" - "THEN"：

指令"IF"表示： 仅当条件表达式成立时，接下来的程序才执行。
指令"THEN"表示“则”。

使用示例：

IF [#600 EQ #601] THEN #602 = 18 *如果 #600 的值和 #601 相同，则 #602 被赋值为 18*

注意："EQ"可被替换为其他关系操作符

条件指令"IF" - "GOTO"：

指令"IF"表示： 仅当条件表达式成立时，接下来的程序才执行。
指令"GOTO"表示前往。

使用示例：

```
IF [#600 EQ #601] GOTO 10
G0 X2 T35
M103 S200
G0 Z6
...
...
N10 G0 X3 T26
```

如果 #600 的值和 #601 相同，那么直接跳到 N10

仅当 #600 与 #601 不同时，这部分程序由数控执行。

注意："EQ"可被替换为其他关系操作符

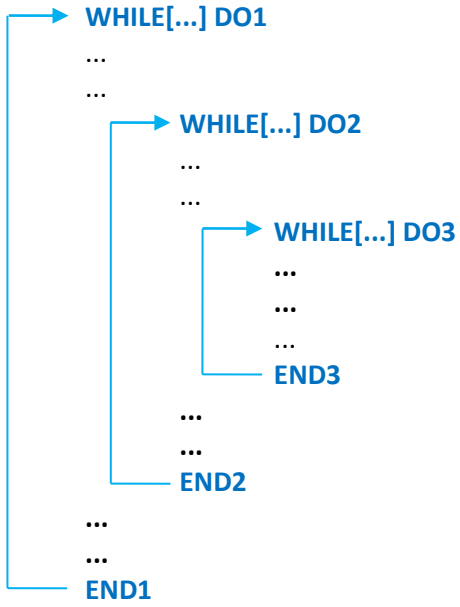
重复指令 "WHILE"：

指令"WHILE"表示循环。
指令"DO"表示进行。

使用示例：

注意："GT" 可被替换为其他关系操作符

内嵌循环示例：



注意：最多三个循环可逐级内嵌。

2.4 子程序

子程序的优点：

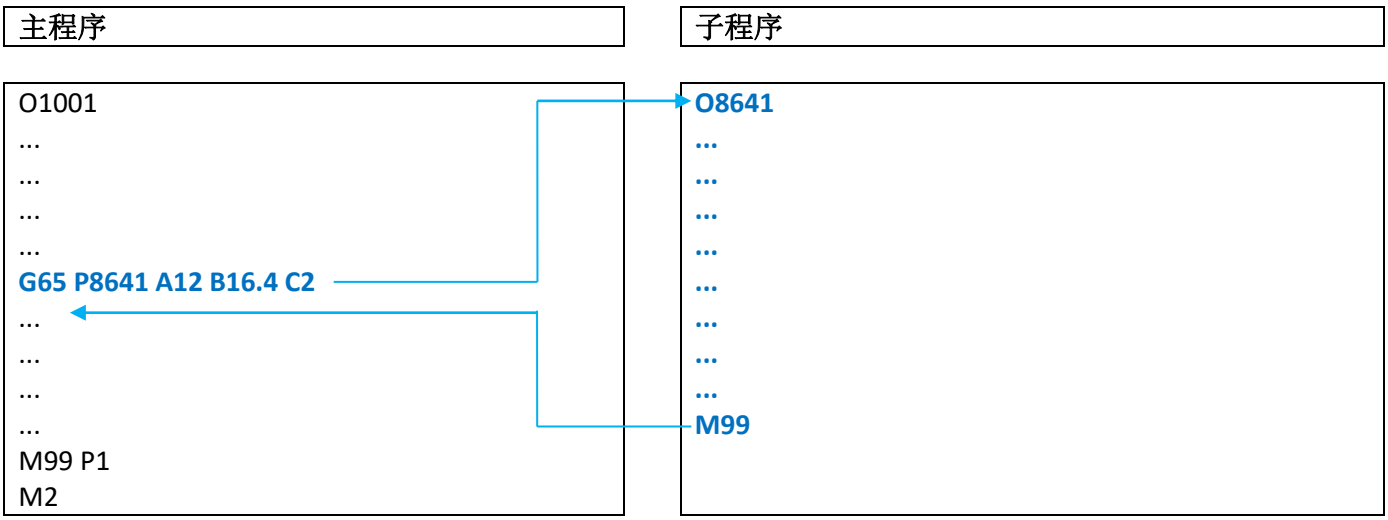
子程序的主要优点是可以重复调用而无需重新编写代码。
因此，它能在单一程序中重复调用，也可在多个不同程序中调用。

子程序的第二个优点是，用户可以在子程序内用参数设置自变量。

子程序调用：

子程序与主程序编码、命名和转移到机床的方法相同。
命名范例"O8641"。
调用子程序可使用功能 G65 Pn {An Bn Cn ...}.

示例：
G65 P8641 A12 B16.4 C2 *子程序 O8641 被调用，自变量 A, B, C 被发送至子程序。*



子程序的自变量：

子程序中发送参数设置自变量是可选的。
如果您希望使用它们，自变量的值根据以下表格中本地变量自动发送。

地址	变量编号		地址	变量编号		地址	变量编号
A	#1		I	#4		T	#20
B	#2		J	#5		U	#21
C	#3		K	#6		V	#22
D	#7		M	#13		W	#23
E	#8		Q	#17		X	#24
F	#9		R	#18		Y	#25
H	#11		S	#19		Z	#26

2.5 显示警报

数控也可显示警报如下：

#3000 = 1 (ALARM)　　*当数控达到该块，则显示警报 "MC3001 ALARM" 并阻止译码。*

使用示例：

```
IF [#600 GE 0] GOTO 10
        #3000 = 2 （负值错误）
N10
```

如果 #600 小于 0，则显示警报"MC3002 NEGATIVE VALUE ERROR" 并阻止译码。

2.6 显示消息

数控也可显示消息如下：

#3006 = 1 (消息)　　*当数控到达该块，则显示消息"MESSAGE" 且不会阻止译码。*

使用示例：

```
IF [#600 GE 0] GOTO 10
        #3006 = 2 （小心负值）
N10
```

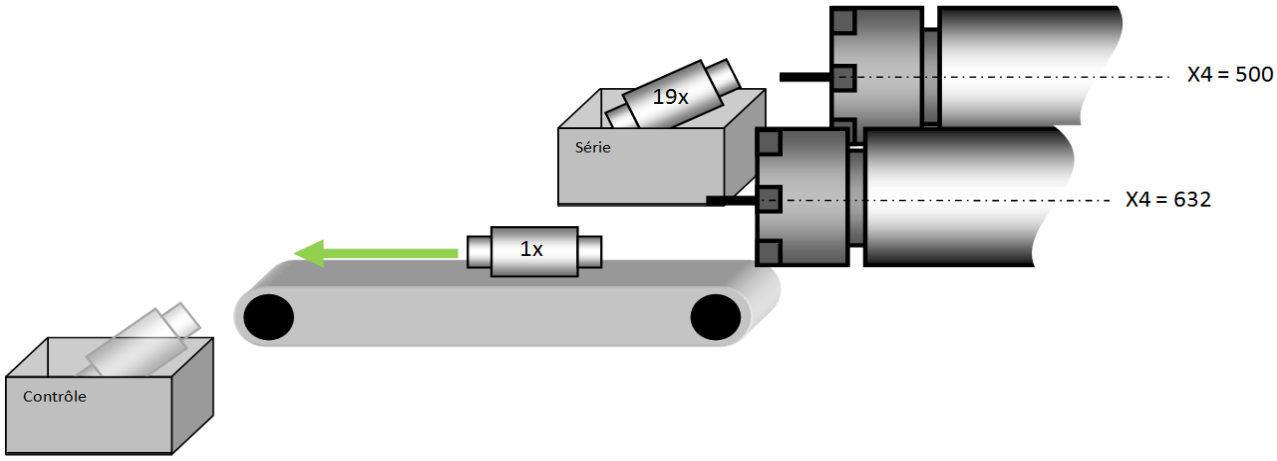
如果 #600 小于 0，则显示消息"CAUTION NEGATIVE VALUE" 且不会阻止译码。

3 具体应用示例

3.1 零件抽样

我们设想一下，您正在生产一系列零件，并需要每 20 个循环进行一次零件检查。
我们看看 B Macro 如何提供帮助。

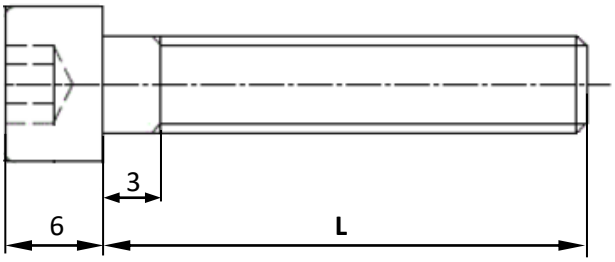
原理：
原理就是已 19 次 将零件弹入机床内的捕捉器，一次置于零件传送带上，所以可以在机床外进行检查。



编程

计数器主轴通道	
#600 = 0	（循环计数器初始化）
#601 = 20	（每循环数要检查的零件）
#602 = 500	（位置 X4 弹出系列零件）
#603 = 632	（用于检查的位置 X4 弹出系列零件）
…	
…	
N1 M120	（循环开始）
IF [#600 EQ #601] THEN #600 = 0	（在第 20 个循环，将计数器复位至 0）
#600 = #600 + 1	（反增量）
…	
…	
IF [#600 EQ #601] GOTO 10	
G53 X500	（不同于从 X500 第 20 次弹出的循环）
GOTO 11	
N10	
G53 X632	（X630 20 次弹出的循环数）
N11	
M84	（弹出）
…	
…	
M121	（循环结束）

3.2 系列零件



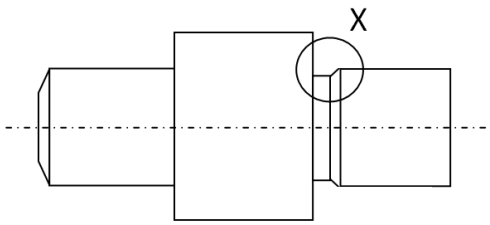
设想您正在生产一系列螺丝。
所有螺丝除了长度"L"外都相同。
生产所有螺丝使用一套程序，而不是每种螺丝一个程序，这很有意义。

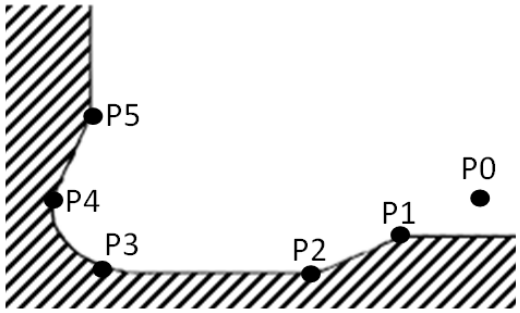
通道 1	
#600 = 53427	（螺丝 ID 号 53426, 53427, 53428, ...）
IF [#600 EQ 53426] GOTO 5	
IF [#600 EQ 53427] GOTO 10	
IF [#600 EQ 53428] GOTO 15	
IF [#600 EQ 53429] GOTO 20	
IF [#600 EQ 53430] GOTO 25	
N5 #601 = 10	（螺丝 53426 长度 "L"）
GOTO 30	
N10 #601 = 12	（螺丝 53427 长度 "L"）
GOTO 30	
N15 #601 = 15	（螺丝 53428 长度 "L"）
GOTO 30	
N20 #601 = 20	（螺丝 53429 长度 "L"）
GOTO 30	
N25 #601 = 22	（螺丝 53430 长度 "L"）
N30	
G800 A10 **B[6+#601] C[#601+2]**	（周期数据：零件长度, C:（零件拾取长度）
...	
...	
N1 M120	（循环开始）
...	
...	
G0 X5 Z2 T12 D0	
G1 **Z-#601** F0.06	（车削间距）
G1 X12	
...	
...	
G78 P020060 Q500 R0.02	（螺纹加工）
G78 X4.2 **Z-[#601-3]** R0 P4300 Q900 F0.7	（螺纹加工）
...	
...	
M121	（循环结束）
...	

这样，你只需要改变程序第一行的 ID 编码就可以改变螺丝尺寸。

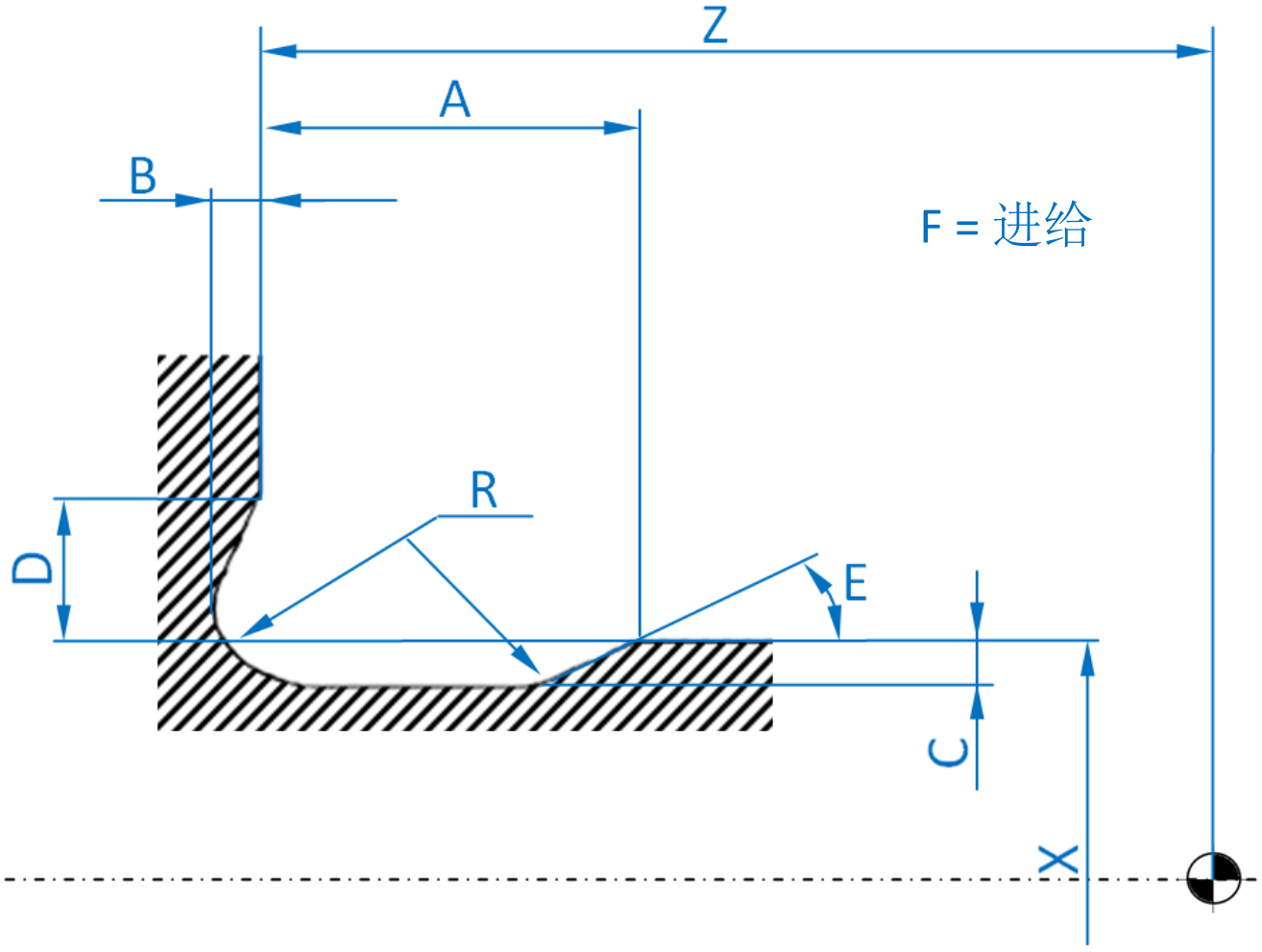
3.3 你自己的加工宏

假设你需要定时为部分零件编写通道，其间需要计算几个标点。为了更省心，你可以编写自己的加工宏。





编写宏：



主程序
... G65 P8512 A2 B0.1 C0.25 D1.15 E30 F0.05 R0.6 X12 Z36　　（子程序 O8512 调用） ...

子程序 **O8512** （加工宏）

```
(*** 变量储存 ***)
#600 = #1                                                    (变量 A)
#601 = #2                                                    (变量 B)
#602 = #3                                                    (变量 C)
#603 = #7                                                    (变量 D)
#604 = #8                                                    (变量 E)
#605 = #9                                                    (变量 F)
#606 = #18                                                   (变量 R)
#607 = #24                                                   (变量 X)
#608 = #26                                                   (变量 Z)
#609 = 2                                                     （安全常数）

(*** 计算 P0 ***)
#611 = #607 + #609                                           (P0 X)
#612 = -[ABS[#608] - #600 - [[#609/2]/[TAN[#604]]]]          (P0 Z)

(*** 计算 P1 ***)
#613 = #607                                                  (P1 X)
#614 = -[ABS[#608] - #600]                                   (P1 Z)

(*** 计算 P2 ***)
#615 = #607 - [#602 * 2]                                     (P2 X)
#616 = -[ABS[#608] - #600 + [#602/TAN[#604]]]                (P2 Z)

(*** 计算 P3 ***)
#617 = #615                                                  (P3 X)
#618 = -[ABS[#608] + #601 - #606]                            (P3 Z)

(*** 计算 P4 ***)
#619 = #615 + [#606 * 2]                                     (P4 X)
#620 = -[ABS[#608] + #601]                                   (P4 Z)

(*** 计算 P5 ***)
#621 = #607 + [#603 * 2]                                     (P5 X)
#622 = -[ABS[#608]]                                          (P5 Z)

(*** ISO 代码 ***)
G90 G95
G0 Y0
G0 X#611 Z#612                                               (P0)
G1 X#613 Z#614 F#605                                         (P1)
G1 X#615 Z#616 ,R#606 F#605                                  (P2)
G1 X#617 Z#618 F#605                                         (P3)
G2 X#619 Z#620 I#606 K0 F#605                                (P4)
G1 X#621 Z#622 F#605                                         (P5)

M99                                                          （子程序退出）
```

3.4 不连续机床清洗

假设您正在生产一系列零件，你需要定时清理机床内部以便弹出切屑。
我们看看 B Macro 如何提供帮助。

原理：
原则就是用高压水枪定时清洁切削工具，但不连续运行清洁程序。
采用这个流程的优点就是降低噪音、减少工厂用电量。
在下面的例子中，我们也借机用空气清洁了计数器主轴夹头。

计数器主轴通道	
#600 = 0	（循环计数器初始化）
#601 = 100	（每循环数清洁机床）
…	
N1 M120	（循环开始）
IF [#600 EQ #601] THEN #600 = 0	（在第 100 个循环，将计数器复位至 0）
#600 = #600 + 1	（反增量）
…	
M11	
M84	（从副主轴夹头上弹出零件）
…	
…	
IF [#600 NE #601] GOTO 10	（每 100 个循环，代码最高运行到 N10）
M532 M11 M1	（阀 1 开启）
M532 M1 M1	（启动高压泵）
G4 X4	（刀具系统 1 清洁）
M532 M11 M0	（阀 1 关闭）
M532 M12 M1	（阀 2 开启）
G4 X4	（刀具系统 2 清洁）
M532 M12 M0	（阀 2 关闭）
M532 M13 M1	（阀 3 开启）
G4 X4	（刀具系统 3 清洁）
M532 M13 M0	（阀 3 关闭）
M532 M1 M0	（关闭高压泵）
M841 M2 M3000	（副主轴中心风机 - 3 秒 用于夹头清洁）
N10	
…	
…	
M121	
…	（循环结束）

3.5 默认机床停止

假设您的机床整周都在生产，但您希望周六晚上停止生产避免工具过度生产零件而导致的磨损，周一早上再重新开始。
当然，对您来说最好不要周六晚上再去工厂停止机床。我们看看 B Macro 如何提供帮助。

原理：
原理就是每个周期都检查数控的日期和时间，并在设定的日期和时间停止机床。

在下面的例子中，我们将停止机床。
在以下日期： 24.06.2017 (#600)
以下时间： 20:35:00 (#601)

通道 1	
#600 = 203500	（停机时间：小时 - 分钟 - 秒）
#601 = 20170624	（机床停机日期：年 - 月 - 日）
…	
…	
N1 M120	（循环开始）
…	
…	
IF [#601 NE #3011] GOTO 10	（检查今天是否为机床停机日期）
IF [#600 LT #3012] GOTO 10	（检查机床停止时间是否已过）
M105	（主轴停止）
M405	
M1105	
M9	（送油结束）
M0	（循环停止）
N10	
M121	（循环结束）
…	

注意： #3011 = 系统变量表示当前数控日期（年月日）
#3012 = 系统变量表示当前数控时间（年月日）

4 最好知道

4.1 循环时间

我们建议您在加工循环前将所有流程都用 B macro 编码（在 M120 之前）。这会最大程度的减少因条件和运算可能造成的循环时间浪费。

4.2 Tornos 培训

也许您会对 Tornos 提供的参数编码培训组合感兴趣，这样您就可以成为真正的专家，从而充分高效的使用这种语言。

4.3 FANUC 说明

FANUC 说明 B-63944 全面介绍了该语言的所有可能用法。