PMPA | PRECISION MACHINED PRODUCTS ASSOCIATION

Nº 131

CRAFTSMAN CRIBSHEET

FANUC Macro Programming Basics

By David Wynn, Director of Technical Services, PMPA

Watch *Dave & Davey video*

Macros are an essential part of high-production CNC programing because they allow for blocks of code to be simplified. They also allow programs to easily be used on families of parts. Macros can even make code more accurate and precise.

Like writing a function in traditional programming, using macro variables in G code can allow a programmer to create repeatable blocks of code that are proven. A macro is a container that holds a value. Using macros changes what the code does but not the underlying process. A programmer must only figure out the details once then let the controller process the math involved in producing the tool path. Let's look at the basic structure of FANUC macros.

Null means no value. It does not mean zero. Think of a variable as a container. When it is null, that means the container is empty.

Range	Type	Persistency*
#1 - #33	Local Variable	Non-Persistent
#100 - #199	Common Variables	Non-Persistent
#500 - #999	Common Variables	Persistent
#1000 and up	System Variables	Persistent

* Persistent variables the value remains after the controller is powered off.
In a non-persistent variable, the value is lost when the controller is powered off.

Some of these ranges vary by controller. Some manufacturers change the address system to match how they utilize the control. This may limit the range of addresses available or change their behavior.

For example: Some controllers change the value of #1-33 values to null after the program has executed. The values for #100-199 are local to the program or channel in use but are persistent until power off. In practice #100 in Channel 1 and #100 in Channel 2 can have different values. In practice, #100 in Channel 1 and #100 in Channel 2 can have different values. (Check your controller's manual to see exactly how your machine works.) I call this "executional persistency." They persist in memory through multiple loops of a program but are volatile because they are reset to null at power off. The values of #500-999 are the same value across the control. Putting a #500=1 allows that value to be called by multiple channels and subprograms and receive the value of "1". This is powerful because values can be shared between subprograms and changed, then pass the new value back to the original program.

The WHILE loop in the example acts as a subprogram and runs through this code until the condition is met, simplifying the code process. Since the code for each flat is combined in this block, a change affects all flats equally. A programmer does not have to go through the code and find all six places and change it manually. This program is utilizing a #100 variable local to this channel and program.

EXAMPLE:
Macros In Use (Milling hex flats)

```
#100=0
#101=.250 (Size of Hex)
WHILE [#100 LT 360] DO1
G0Y.888Z.3125
X#101
M18C#100
G1Y-.888F7.638
#100=#100+ 60
M18C#100
G1Y.888
#100=#100+ 60
END1
G0X.6
M60M5
G99
```

Macros are a powerful way to level up a CNC program. Using advanced programming techniques allows programmers to simplify complex tasks. Once a block of code is designed, it can be utilized on multiple programs. Code reusability is key to accurate and precise programming. Humans make mistakes but when we use proven systems (reusable code for example), we increase our ability to perform.