Journal of Material Science and Mechanical Engineering (JMSME)
p-ISSN: 2393-9095; e-ISSN: 2393-9109; Volume 3, Issue 3; April-June, 2016 pp. 254-257

http://www.krishisanskriti.org/Publication.html

A Practical Approach to the use of Local Variables in CNC Machines Programming for Fanuc Custom Macros

Mohd Asif Hasan

Department of Mechanical Engineering, University Polytechnic, Faculty of Engineering and Technology, Aligarh Muslim University (AMU), Aligarh - 202002 (India)
E-mail: hasan_in@hotmail.com; asif.hasan.bp@amu.ac.in

Abstract—*The capabilities of Computer Numerical Control (CNC) Machines are well-appreciated but the challenging aspect is Programming of the CNC Machines. Ever since the advent of CNC Machines, constant efforts are being made to ease and simplify the programming methods and procedures of CNC machines. One such effort is in the direction of development of Macros for repetitive type of tool motions. Macros are simple part-programs which reside in the memory of the controller and are called using a specific code for the macro. All sort of canned cycles are basically Macros. Canned cycles for turning, threading, stock removal, drilling, etc. are now-a-days provided as standard feature available on CNC machines which simplifies the CNC programming for these operations to a great extent. However, development of canned cycles or macros, as per the requirement of the user, requires an in-depth knowledge of Manual Part Programming (based on G and M codes) and Macros (part-programs based on Variables). In Fanuc systems, the programming language or environment available for the development of macros is known as "Custom Macro B". As development of Macros is primarily based on the use of particular type of variables know as Local Variables, this paper presents a conceptual framework of these variables for the development of macros in Fanuc's Custom Macro B environment.*

Keywords*: Computer Numerical Control; CNC; CNC Programming; Macro; Parametric Programming; CNC Variables; Local Variables; Custom Macro B*

1. INTRODUCTION

Computer Numerical Control (CNC) machines are those machines which are controlled by numbers while making use of computers for processing the information fed to these machines. The numbers used for controlling these machines are actually alpha-numerals and are popularly known as Codes of CNC Machines. These codes are arranged in a logical sequence depending on the type and sequence of operations to be performed on a job. This logical sequence of CNC codes is known as Programme of Instructions or simply Part Program. These part programs are fed to the Controller which is the brain of the CNC machine, for processing and execution of these instructions. If these part programs are developed by the operator or human being, then this method of programming the CNC machines is known as Manual Part Programming. However, this method of programming the CNC machines is tedious and bounded by the limitations of human being, but presently is almost inevitable as CNC machines only understands this format of G and M codes. But, there are other methods of programming the CNC machines, like Conversional programming and programming using CAD/CAM softwares, which basically helps the human being or operator of these machines to generate the part program in this standard format (based on ISO 6983 and others) [1, 2, 9].

Another method of CNC Programming is combining the Manual Part Programming with the Macros which provides enormous advantages and ease of programming in many situations. Macros often serve as a special solution to special requirements. Although CAD/CAM programming systems have become very popular and are on the rise, they do not and cannot always replace macro programming, for various reasons. Macros are simple part-programs which reside in the memory of the controller and are called using a specific code for the macro. All sort of canned cycles are basically Macros [1, 2, 3].

CNC programming using Macros is referred to as Macro Programming, Parametric Programming, or simply Macros. CNC programming using Macros can be compared to any computer programming language like BASIC, C Language, and PASCAL. However, this programming language or feature is available right in the CNC controller and can be accessed at G code level, which means that it can be combined with manual programming techniques of CNC machines [4]. As it is like any computer language and thus also possesses Computer-related features like variables, arithmetic, logic statements, and looping. Like computer programming languages, this feature of CNC programming using Macros also comes in several versions. The most popular is Fanuc's Custom Macro B (used by Fanuc and Fanuc-compatible controls). Others include User Task (from Okuma), Q Routine

(from Sodick), and Advanced Programming Language [APL] (from G& L). In addition to computer related features, Custom Macro B could also be used for extensive CNC related features which are more commonly related to the utilities such as part counters, tool life managers, etc. and driving accessory devices such as probes, in-process and post process gauging systems, etc. [5,6,7]. A Part Program developed using Macros for one control model may not work with another control model. However, the logic and general approach discussed here can be adapted for various models of Fanuc controller and also for control systems other than Fanuc.

2. MACRO PROGRAMS AND SUB-PROGRAMS

In simple words, Macro programming is programming the CNC machines in standard G and M code format while making use of the capabilities of the controller in making arithmetic calculations, feeding formulae, storing, reading and assigning values to the system variables (with certain restrictions) as well as Local and Common variables, etc. These variables could only be numbers with associated fixed function and functionally are just like algebra variables. They can be assigned values, and when referred give back the last value they were assigned. [1, 2].

Development of Macro programs requires the deep knowledge of the *Preparatory Commands* (G-codes) and the *Miscellaneous Functions* (M-codes) in a part program. A macro program developed using Custom Macro B or else resembles a standard CNC program to a certain extent, but includes many features not found in regular programming. Essentially, a macro program is structured as a regular subprogram. It is stored under its own program number (**O-**), and it is called by the main program or by another macro, using a G-code (typically **G65**). However, in a very simple form, macro features can be used in a single program as well, with out the macro call command [3].

Use of macros becomes essential in some areas of CNC programming irrespective of the method of programming, whether a manual method or a CAD/CAM method is used. Some of the areas where CNC programming using Macros (or simply Macro Programming) is indispensable or at least offer an edge over other methods are Part Programming for Family of Parts, developing Special G and M codes, Canned Cycles and Complex tool motions, Offset Control, Probing and Gauging.

Although, Subprograms are the first logical step into the macro development, the major difference between the two unique programming methods is the flexibility macros offer. Over all, following are the three agenda items that are the most significant in programming macros: Variable data input, Mathematical functions and calculations, and Storage and retrieval of current machine values. Only the first agenda item i.e. variable data input is discussed in this paper [3].

3. RELEVANCE OF VARIABLES IN MACRO

The most noticeable feature of Macros is the use of Variables for storing numerical values. A variable is a mathematical quantity that can assume any value within its allowed range and format. The word variable means change or changeable. As the data that may change is stored in the variables forms the basis of flexibility in Macros. In macros, variables can be used instead of real numerical values and they can be treated like algebraic variables for various mathematical operations, for example, by adding two variables together, to get yet another value.

The syntax for a variable is the pound sign followed by a number which may be of as many digits as the machine's controller supports to identify the variable. For example, it may be written "#1 = 10.0" to assign the value "10.0" to the variable "#1". These variables are of great use when you might want to change a value of these variables in different situations resulting in all other values based on these variables will be updated on its own. This feature of Variables add flexibility to the macro program but also benefit from other features, such as input data integrity, allowable range checking, etc.

The basic rules governing the declaration of variables is that a variable must be defined first, and only then it can be used in a program or a macro, for example, #5 = 20 indicates that the variable number 5 is assigned or defined with a Value of 20. Variables can also be defined by using an *expression*, where the expression is typically a mathematical formula or a general calculation [3, 10]. Expressions must be enclosed in square brackets, for example:

#7= #7 * [#5 + #6]

where the brackets force calculation of #5+#6 to be performed first, before being multiplied by #7.

Any complex calculations can be nested within square brackets which always follow the standard mathematical hierarchy relating to the order in which calculations will be processed. When performing mathematical calculations, the *type* of every numerical value is important. In simple terms, real numbers are typically used for calculations, whereby integer numbers are used for counting and other applications that do not require a decimal point. Moreover, to avoid any chance of error, a variable that is defined in the macro program body must always be entered with the decimal point for all dimensional values, such as position locations, distances, feed rates, or any other definitions that use metric or English units.

Once a variable is defined, it can also be used by preceding it with the desired Fanuc program related address (character), which is a capital letter of the alphabet, such as F, S, G, M, etc.

Variables in Custom Macro B are fixed and designed meticulously to obtain maximum leverage. For a better understanding of variables, it is important to understand the various types of variables, their differences and respective applications. In Custom Macro B, there are the following four different categories of variables, called the variable types: (a) NULL variable (#0) (b) LOCAL variables (from #1 to #33) (c) COMMON or Global variables (from #100 to #149 and from #500 to #531) and (d) SYSTEM variables (#1000 and up). However, the emphasis of this paper is on Local Variables.

4. USE OF LOCAL VARIABLES IN MACROS

Local variables are called local as their stored values are only applicable to the macro in which they are defined and are non-transferable between macros which sometimes present a serious challenge to the macro developer.

LOCAL variables are only temporary and are used in a macro body to hold certain data. When the macro is called, the local variables are set to their assigned values. When the user macro is completed and exits (using the miscellaneous function M99 or else), or the control power is turned off, all local variables are set to null values, i.e. they cease to exist. The local variables are normally cleared, called Purging, either through the control panel or a program code. The local variables can be purged by pressing the control Reset key or external Reset key or Emergency button. The local variables also get purged when they encounter Program End code (M30) or Sub-Program End code (M99). The local variables could also be purged by equating them with Null Variable (#0) after their use in the Macro program itself. As it is the Miscellaneous Codes (M30 or M99) that clears the local variables and not a jump from one program to another, the feature of Nesting could be utilized for defining and redefining local variables up to five times i.e. once in the main program and once for each macro level up to four levels as only four levels deep macro nesting is allowed. As after every jump in the nesting of macros the newly defined local variables take over but the old set(s) still remains in the memory and takes control when macro returns back to that level, which needs to be cleared by making an appropriate use of M30 and M99.

Each local variable is associated with an assigned letter of the English alphabet which makes value associated with the alphabet(s) in the macro call argument specified in the main program be transferred to the respective local variable number in the body of the macro program. There are two options available for the use of Local Variables, Assignment List 1, which has 21 local variables available, and Assignment List 2, which has 33 local variables available [3, 8]. Following is the assignment list 1 and assignment list 2 as defined by Fanuc expressing local variable number to be used in the body of the macro program and its corresponding Argument Address (English alphabet with/without numeral as suffix) to be specified for expressing user supplied values for these local variables in the argument of the macro call:

Assignment List 1: A (1), B (2), C (3), D (7), E (8), F (9), H (11), I (4), J (5), K (6), M (13), Q (17), R (18), S (19), T (20), U (21), V (22), W (23), X (24), Y (25), Z (26).

Assignment List 2: A(1), B (2), C(3), I1 (4), J1(5), K1(6), I2(7), J2(8), K2(9), I3(10), J3(11), K3(12), I4(13), J4(14), K4(15), I5(16), J5(17), K5(18), I6(19), J6(20), K6(21), I7(22), J7(23), K7(24), I8(25), J8(26), K8(27), I9(28), J9(29), K9(30), I10(31), J10(32), K10(33).

On a careful observation of the Assignment list 1 and assignment list 2, it can be found that the following 12 local variables are missing from assignment list 1: #10, #12, #14, #15, #16, #27, #28, #29, #30, #31, #32, #33. But, while using assignment list 1, these 12 variables can also be defined internally as local variables within the macro body only. The difference between these 12 variables and the 21 variables of the assignment list 1 is that these 12 variables are not tied up to a letter address. On a further careful observation of both the assignment lists, it can be found that the following five alphabets are missing: G, N, O, P, L. These five are restricted alphabets and cannot be assigned any value for any purpose as G is used to address Preparatory Commands, N for Block Number, O for Program Number, P for Program Number Call and L for Number of Repetitions of the macros, subprograms or canned cycles. Of the five, only the letter G can be used for a special purpose, such as a definition of a new G-code.

The following illustrative sample main program and macro will clarify the use of Local Variables from the Assignment List 1:

```
O1234 (MAIN PROGRAM)
N1 G21 G90
N2 ….
N3 ….
…….
N10 G65 P8080 X 10.0 Y20.0 Z40.0
N11 ….
…….
N40 M30
%


O 8080 (MACRO)
N1 ….
N7 G00 X#24 Y#25 Z#26
N8 ….
N20 M99
%
```

In the sample main program, a macro call has been given using G65 code to the macro saved with number 8080. The arguments of the macro are X, Y and Z with user specified values of 10.0, 20.0 and 40.0 respectively. These user supplied values of X, Y and Z shall be transferred in macro (O8080) to the corresponding local variable numbers of X, Y and Z which are #24, #25 and #26 respectively.

In the assignment list 2, the suffix of each set of I-J-K specifies the assignment order for the argument set defined in **G65** macro call but the order has to be followed very strictly and carefully, for example, in G65 macro call the first mentioned I represents variable #4, the second I represents #7, the third I represents #10, and so on and accordingly for J and K. It is because of this order confusion of I-J-K set that assignment list 2 is rarely used.

5. DISCUSSION

In order to make efficient and effective use of CNC machines, knowledge of the development of macros is a must. As macros are heavily dependent on the use of variables, the first step towards the development of macros is to understand the structure of Variables. In this paper, the structure of Variables is discussed in the Fanuc's "Custom Macro B" environment. Moreover, a conceptual framework has been developed for a particular type of variables known as Local Variables which are primarily used for the development of Macros. Unless the features, capabilities and limitations of these Local Variables are well understood, an efficient and highly potential macro can not be developed. Thus, this paper also focussed on the features, capabilities as well as limitations of these Local Variables. The future research direction which is also the limitation of this paper could be the practical implementation of this approach for the development of macros for required tool motions.

REFERENCES

[1] *Hasan, M.A. (2015),* "Computer Numerical Control Machines: An Account of Programming Methods and Techniques", *Journal of Material Science and Mechanical Engineering (JMSME),* Vol. 2, No. 12, pp. 14-17.

[2] *Hasan, M.A. (2015),* "Computer Numerical Control Machines: An Introduction to Parametric Programming using Custom-Macro B", Journal of Basic and Applied Engineering Research, Vol. 2, No. 18, pp. 1566-1569.

[3] *Smid, P. (2005), "Fanuc CNC Custom Macros",* Industrial Press, Inc., 200 Madison Avenue, New York, NY, USA.

[4] Djassemi, M. (1998), "A Parametric Programming Technique For Efficient CNC Machining Operations" *Computers and Industrial Engineering,* Vol. 35, No. 1-2, pp. 33-36.

[5] *Lynch, M., "Modern Machine Shop Magazine",* www.mmsonline.com

[6] CNC Concepts, Inc., *www.cncci.com*

[7] www.cnccookbook.com

[8] http://www.machinetoolhelp.com/Applications/macro/macro_variables.html

[9] Adithan, M. and Pabla, B.S. (2007), "*CNC Machines",* New Age International (P) Limited, Publishers, New Delhi.

[10] Nikiel, G. (2007), "Computer-Aided CNC Programming for the Machining of Non-Typical Parts", *Advances in Manufacturing Science and Technology*, Vol. 31, No. 4, pp. 21-36.