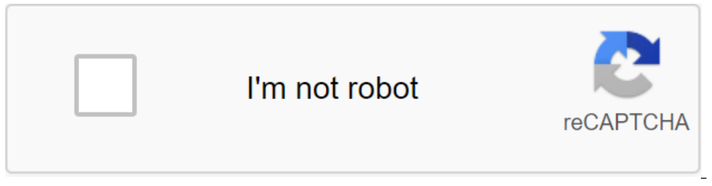

Continue

Fanuc macro variables

Fanuc Macro System Variable System Variable Lists Parametric Programming Target Variable Number Series Variable Number #1000?#1015 #1032 #1100?#1115 #1132 #1133 #2001?#2064 #10001-#10999 #11001-#11999 #2001-#2200 (Offset No.1?64) Geometry Offset Offset Value (No.1?49) Offset Offset Value Wear (Offset No. 1?99) Geometry Offset Value (Offset No.1?99) #2201-#2400 #10001?#10400 #11001?#11999 #12001?#12999 #13001?#13999 #3000 #3001 #3002 #3003 #3004 #3005 #3007 #3011 #3012 #3901 #3902 #4001?#4022 #4102 #4107 #4109 #4111 #4113 #4114 #4115 #4119 #4120 #4130 #5001?#5008 #5021?#5028 #5041?#5048 #5061 #5068 #5081?#5088 #5101?#5108 #2500 #2600 #2700 #2800 #2501 #2601 #2701 #2801 #2502 #2602 #2702 #2802 #2503 #2603 #2704 #2804 #2505 #2605 #2705 #2805 #2506 #2606 #2706 #2806 #5201?#5108 #2500 #2600 #2700 #2800 #2501 #2601 #2701 #2801 #2502 #2602 #2702 #2802 #2503 #2603 #2704 #2804 #2505 #2605 #2705 #2805 #2506 #2606 #2706 #2806 #5201 ???#5208 #2803 #2504 #2604 #5221???#7028 #7941???#5228 #5241???#5248 #5261???#5268 #5281 #5288 #5301???#5308 #5321???#5328 #7001?#7008 #7021 #7028 #7941?#7948 #14001?#14008 #14021?#14028 #19980?#19988 Signal Interface Interface Output Signal Interface X axis displacement tool nose radius compensation Imaginary tool tip position Y axis offset tool compensation (memory displacement A) harvesting zero point offset value G54 blank zero point displacement value G55 Tool compensation (memory B displacement) #5041?#5048 #5061?#5068 #5081?#5088 #5101?#5108 #2500 #2600 #2700 #2800 #2501 #2601 #2701 #2801 #2502 #2602 #2702 #2802 #2503 #2603 #2704 #2804 #2505 #2605 #2705 #2805 #2506 #2606 #2706 #2806 Alarm Clock Control of one stop block, Wait for signal for THE CHANNEL, accurate stop check Installation Status Mirror Image Watch Compensation Tool (Memory Displacement C) Number of parts Modal Information Unit End Position (Working Coordination) Machine coordinates Skip Signal Position (Working Coordination) Tool of Value Displacement Servo Position External Blanking zero point offset value G54 zero point #2701?#2749 #10001?#10099 #15001?#15099 #2101?#15099 #2101?#15099 #2101?#15099 #2101?#15099 #2101?#15099 #2101?#15099 #2101?#2164 #2801?#15099 #2101?#15099 #2101?#2164 #2801?#15099 #2101?#15099 #2101?#15099 #2101?#15099 #2101?#15099 #2101?#15099 #2101?#15099 #2101?#15099 #2101?#15099 #2101?#15099 #2101?#2164 #2801? #2849 #11001?#11099 #16001?#16099 #2201?#2264 compensation value of the G57 harvesting zero point offset the value of the G58 blank zero point #2901?#2969 #12001?#12099 #17001?workpiece zero point offset the value of the G55 #14099 #19001?#19099 #2001?#2200 the blank zero to compensate for the value of the G56 zero point offset value of G59 #17099 #2301?#2364 #2301 #2364 #2301?#2364 #2301 ?#2364 #2301 #2364 #13001?offset value G54.1 P48 blank zero point offset value G54.1 P1 blank zero point value G54.1 P2 blank zero offset value G54.1 P48 harvesting #13099 #13001?#13099 #2401?#2449 #2451?#2499 #14001?#10001?offset value G54.1 P2 work zero #10999 #2001-#2200 #2201-#2400 offset value of the G57 blank zero point offset value of the G58 harvesting zero point offset value G59 blanks zero point G54.1 P1 blanks zero point zero point offset offset value in Fanuc Macro variable system, parametric programming and Fanuc Macro Macro variable system, Parameter It all sounds pretty secret, doesn't it? In fact, it is not. They are just buzzwords and jargon that stand behind some pretty simple concepts. Mastering them is pretty easy and they will give you tremendous power and control in your G-code programming. The following few lessons in the tutorial stand together, like a mini tutorial on macro programming and Fanuc Macro B. In many ways, macroprogramming is the highest level of G-code programming. It offers the greatest flexibility and greatest potential power of any of the G-Code programming methods. Without macroprogramming, G-Code isn't really a full-fledged computer language, it's more of a recording of a series of manual steps. This is useful, but computers and CNC controllers can do much more. The word about Fanuc Macro B Fanuc Macro B is by far the most common macro programming dialect. Not all controllers support a full Macro B, and there are variations supported by some non-Fanuc controllers. We can't document every G-Code dialect here, but the concepts offered in Macro B will be similar to what you see elsewhere, and therefore apply to these cases with slightly different syntax and possibilities. If you don't have a controller with a Macro B, it's still worth checking it out. Just use our G-Wizard CNC Simulator and Editor to play with Macro B capabilities. Parameterized programming So far, all of our g-coding has been done without any reference to variables or formulas. If we wanted to go to a certain coordinate, we had to enter this coordinate, absolute or relative, to get there. A huge amount can be achieved in this way, but just as algebra is more powerful than arithmetic, so parametric programming is more powerful than the basic G-code, and for the same reasons. Once you have the variables, you can start parameterizing programming. You'll be able to create general destination procedures that use settings to link them to specific uses you want to make. Imagine being able to create your own bespoke canned cycles that can do just about anything. Custom drilling cycle, or perhaps routine is automatically the serial number of your parts. Almost anything is possible. Covering the entire parameterized will include three chapters in our tutorial. In this chapter we will deal with variables and formulas. The next chapter will delve into how to divide the code into routines and access the routines. Finally, we'll deal with macro-control macro-control Not all controls support parameterized programming. We'll be dealing with the Fanuc Macro B dialect for it, and we'll throw in mention of Mach3 and LinuxCNC's parameterized programming features as well. Options for customizing the machine for the controller (aka Why we're going to quit talking Options) you may have noticed that different companies often make a CNC controller compared to the actual machine. In fact, this is almost always the case. There is a collection of settings inside the controller, which are called settings that are used to perform this controller configuration to the machine. For example, travel, spindle speed range, speed of rapid passage and many other types of information that are crucial for the control to work properly with the machine. For this reason, you don't want to go poking around with the settings too much, or you can scramble some of these settings. Reading a system variable is ok, but don't assign them any values if you don't know for sure what they're doing. You need to make sure that you have a good backup of all your settings in case the battery fails on the controller or they get inadvertently altered. Usually there is a procedure available for your controller to use the DNC to perform backup. Now here's the tricky part: Options and Macro Variable are two different things! I don't know why we would call it Parameterized programming and then immediately quit talking about parameters that have little to do with parameterized programming. It's tempting to think the settings are just system variables (more on that system variable at the moment), but they're completely different. There are ways to set some settings from part of the program, but this is not the most common way to deal with them, usually it is done through a control panel. So we'll stick to the term variables and leave the parameters for the future chapter. What are variables? First, what are variables? Simply put, they are just like algebra variables that you can use in your G-code. They may be assigned values, and when you refer to them, they return the last value they were assigned. A syntax for a variable is a pound sign, followed by a number of to how many digits the controller that identifies the variable supports. For example, we can write #1 and 100 to assign 100 variable #1. There's even a special zero value that says the variable has never had a value assigned. The variable #0 has zero value, and you can give any other variable zero value by simply assigning #0 to it. For example: #100 and #0 (Make #100 zero) These variables fall into the that affect their value and how they can be used. Here's how Fanuc thinks about these ranges, for example: #0 Null you can never assign a assign For #0, this value is always zero, which means no value. #1 - #33 Local variables are used to transmit arguments on macros and as temporary scratch storage. The controller won't remember the value of local variables when power outages. Local variables nest when used with sousprograms, so make sure you understand how it works. #100 - No. 199 #500 - #999 common variables share all your macro programs. When the power is turned off on the controller, the #100 - #199 is cleaned to nothing. #500 - #999 remember their values the next time power is switched on. #1000 and up system system systems systems can be used to tell things about what the controller is doing, such as the current position. Don't assign anything to these if you don't know what they're doing! Fanuc Variable Macros: Please note that these ranges can vary depending on controllers and especially for controllers that do not appear to be Fanuk! Mach3 has 10,320 variables, from #0 to #10320. There's not exactly variety found in Fanuc, but still, a lot of system variables are available, so you can get on with things like work bias. The Mach3 variables are saved from one control load to another, so be sure to reset them if you don't expect them to matter. What are variables good for? Use variables whenever you think you can change value in different situations. For example, suppose you have a macro or routine that cuts a square pocket. You probably want to have variables that allow you to determine the size of the square and the channels and speeds so that they can change as the material changes. Depending on the routine, you may need a variable for the top left corner of the square, or you can simply let that corner be the current position when called a routine. You may need a variable that determines the depth of a square pocket. Finally, you may need a variable to determine the diameter of the instrument. What variables should I use in my programs? System variables and local variables have some special behaviors, so stay away from them if you specifically want these behaviors. System variables refer to certain things that occur in the controller, so they cannot be used as general purpose variables. Local variables demonstrate nest behavior with macros, so wait until you read about macro programs and understand them before deciding to use a local variable. Common variables are there for adoption, so try to stick to using these variables for a general programming purpose. When variables cannot be used most addresses (remember address of the word?) can be variables, but not all. For example, I can't have a sequence number that is variable. NC10 is not allowed. Here's a short list not allowed: - Program numbers: O'10 is not allowed. Not :#10 for controls that allow : for program program - Sequence numbers: N-10 is not allowed. - Block Skip Address: /1 is normal, but /#1 is not allowed. -- WHILE.. Do. END Addresses: DO1 is allowed, DO-1 is not. fanuc macro variables list. fanuc macro variables programming. fanuc macro variables control commands. fanuc macro system variables list pdf. fanuc oi macro variables. fanuc 30i macro variables. fanuc macro system variables list