CNC Programming: Friend or Foe to Productivity?

Our objective is to promote thoughtfulness and understanding in the area of CNC programming – why we should do certain things and why we need to give up our bad habits. We will do so by going through a series of techniques that highlight the dos and don'ts, the highs and lows, and the good, the bad, and the ugly. Most techniques will in some way improve CNC machine utilization.

Objectives of any CNC program 4
Compatibility and consistency 4
Appropriate application of new/unique CNC features 4
Your measurement system choice 5
Units of value specification 5
Least input increment 5
Tool path precision 6
What happens when you switch measurement systems? 6
Making programs for similar machines compatible 7
You say your turning center has only one spindle range? 9
Determining which machine is being run from within CNC programs 9
Maintain consistency in program structure 11
Maintain consistency among programs: Cutter radius compensation variations 13
Method one: programmed path is work surface and offset is the cutter's size 13
Method two: programmed path is cutter centerline path and offset is the deviation from planned cutter size. 14
Which method is better? 14
Be consistent! 15
Maintain consistency among programs: Tool length compensation variations 15
Ease of use 15
Documenting in the program 16
Program headers 16
Tool/process names and current size specification 17
When doing something out of the ordinary in the program 18
When making program changes 19
Simplify setup-related tasks: Program zero assignment 19
Eliminate Program Zero Assignment Tasks with Qualified Setups 19

Eliminating the measurements required to assign program zero with qualified setups 20
What about crashes? 20
Eliminating the fixture offset entry 20
Facilitate the measurements required to assign program zero with a spindle probe or edge finder 21
Centering the Y axis for CNC lathes 22
Simplify setup-related tasks: Cutting tools 24
Facilitate tool offset entries with a program created by tool length measuring device 24
Facilitate tool length measuring on the machine with a tool length measuring probe or a program 25
Do not make setup people enter tool nose radius compensation offset values on turning centers 26
Simplify setup-related tasks: Program verification 28
For machines that allow it, ensure cutting tools can be run 28
Provide obvious restart block for machines that allow operators to rerun cutting tools 29
Program trial machining operations 29
A test part program for sliding headstock lathes 31
Deal with raw material Z surface variations in your machining center programs 35
Deal with raw material X/Y surface variations in your machining center programs 37
Simplify production run tasks 39
Automate sizing adjustments for long running jobs 39
Keep track of how long tools last before the must be replaced 41
A program that tells operators when tools will get dull 43
Eliminate calculations needed to make sizing adjustments 45
Make it easy for operators to determine which measuring device to use and which offset to adjust 48
Safety 49
Machine safety 49
Not knowing your M-codes could lead to machine longevity issues 49
Applications for M codes 50
Some rather obscure M codes 50
Confirm all initialized states 51
Include more than just initializing G codes 51

An example 52
Use the best “hand” of tooling when performing powerful machining operations on turning centers 53
Error trap your CNC programs 54
A pallet changer testing program 55
An example: 56
When to limit spindle speed on CNC turning centers 58
NEVER program around machine problems 59
NEVER let that bar hang out – and other safety issues 61
Efficiency 62
Structure your programs to minimize program execution time 63
Efficient tool changing / turret indexing 63
Machining center suggestions 63
Where is the tool change position? 63
Get the next tool ready 63
Orient the spindle on the tool’s retract to the tool change position 63
Load tools sequentially for short cutting operations 64
Look for machines that allow tool replacement during automatic operation 64
Turning center suggestions 64
Program Constant Surface Speed Efficiently 66
Program spindle range changes appropriately 67
1) Know your spindle’s power and speed characteristics 68
2) Know how long it takes to change spindle ranges 68
3) Know how to select spindle range 68
4) Know how spindle speed can affect cycle time 69
5) Know when to change spindle ranges 69
Reduce air cutting time when appropriate (rapid approach distance and feed off amount) 69
Check parameters that affect cycle time 71
G73 chip break peck drilling cycle 71
G83 deep hole peck drilling cycle 71
G71 rough turning cycle 71
Where do you find parameter documentation? 71

Objectives of any CNC program

We begin with four objective you should strive to achieve with every CNC program you create:

1. Consistency and compatibility
2. Ease-of-use
3. Safety
4. Efficiency

While some objectives may conflict with other, most of the suggestions we make will have a positive impact on productivity.

Compatibility and consistency

For our purposes, we describe consistency as maintaining the same program structure for all programs – related to a given machine – and when possible among similar machines. Indeed, CNC operators want to see the same words and commands in the programs they run. This leads to familiarity and helps operators know when something is wrong.

Additionally, we want to ensure consistency among the various ways certain CNC features can be used. Some CNC features can be used in multiple ways. And variations among programs often requires operators to do things differently from one program to the next – causing wasted time and increasing the potential for mistakes.. Inconsistency here is often the result of having multiple programmers program a machine over the years. Some handled a feature one way, while others handled it another.

Compatibility, by comparison, is having the ability to run the same program on multiple similar or identical machines. This minimizes the need to have a different program for each machine on which a component is produced.

Appropriate application of new/unique CNC features

This is a tough one. The longer your company has been using CNC machines, the bigger the issues you probably experience here. CNCs have been evolving for well over forty years – and the list of improvements and new features is lengthy.

CNC manufacturers have always kept their machines *backward compatible*. Programs written for older CNCs continued to run properly on newer CNCs. On the good side, this maintained a high degree of compatibility from older machines to newer machines. Users didn't have to recreate programs every time a new machine (having a newer CNC control) was purchased.

But on the bad side, CNC users often found themselves using very old techniques on very new controls. Examples include programming numerical values in fixed format when decimal point programming became available (X100000 instead of X10.0 for an X coordinate of 10-inches). If the

user decided to use decimal point programming for the newer machine/s, those programs would not run on older machines requiring fixed format. In some cases, users continued with old techniques for new machines even after the older machines were gone.

This is but one example of the countless improvements that force CNC users to decide: Do you maintain program compatibility – or do you take advantage of newer/better features when they become available.

This trend continues to this day. There are many five-axis CNC users (predominantly in the aerospace industry) who continue to use 20+ year old methods for sculpturing surfaces when switching to newer features (on the machines that allow it) could cut program execution times in half.

Admittedly, this is a tough nut to crack – and I don't have much of a suggestion. But it is important to know that this trend exists. You should regularly reevaluate your programming methods. At the very least, when an older machine is replaced, consider updating your programming methods. And remember, older programs will of course still run on the newer machines.

Your measurement system choice

Most CNC users have already chosen a measurement system (Imperial or Metric) and will use it exclusively for their CNC machine tools. The choice was probably an automatic one, based upon which measurement system they were using when they purchased their first CNC. In the United States, many companies have traditionally used the Imperial measurement system. Companies with an Asian or European influence, on the other hand, have traditionally used the Metric measurement system.

When the company happens across a workpiece dimensioned with the other measurement system, they simply convert the print to their measurement system of choice and run the job in that measurement system.

Units of value specification

In the Imperial system, the most basic unit is the inch. In the Metric system, it is the millimeter. This unit will apply to axis and related specifications (like X, Y, Z, R, I, J, and K).

Feedrate specification in the Imperial system is done in inches per minute or revolution. It is specified in millimeters per minute or revolution in the Metric measurement system. Spindle speed in constant surface speed mode (for turning centers) is done in surface feet per minute in the Imperial system and in meters per minute in the Metric system.

Thread pitch – which of course, affects feedrate – is specified in threads-per-inch in the Imperial system (requiring a calculation to determine the actual pitch). In the Metric system, pitch is specified directly – in millimeters.

Least input increment

The least input increment (smallest programmable value) is usually 0.0001 inch in the Imperial system and 0.001 mm in the Metric system, though there are high precision machines that allow 0.00001 inch and 0.0001 mm.

Think about the dramatic difference in resolution here. 0.001 mm is less than half of 0.0001 inch – actually, 0.001 mm is 0.000039 inch, or 39 millionths of an inch. This gives the Metric measurement system as it is applied to CNC machine usage a much better resolution than the Imperial measurement system. It often allows the mean value of tolerance bands to be more precisely specified in the program – and it provides more offset settings within a given tolerance band.

For example, a 0.001 inch overall tolerance has about ten offset positions in the Imperial mode that will put the machined attribute within its tolerance band. This same tolerance will have about 25 offset positions in the metric mode. This, again, provides finer resolution for offset adjustments – which can be especially important when holding very small tolerances. When performing close-tolerance work on workpieces dimensioned in the Imperial system, some companies will actually convert the dimensions and tolerances to the Metric system and run the part in Metric – just to gain this offset setting resolution advantage. It allows them to do higher precision work.

Tool path precision

With many CNC machines, the resolution advantage of the Metric measuring system carries over to the multi-axis motions. Within any interpolated motion (like linear or circular interpolation), the machine will internally divide the motion into a series of very tiny single-axis steps. The step-size – again, for many CNC machines – is related to the currently instated measurement system. These machines will actually follow a more precise tool path when in the Metric measurement system.

Don't believe it? Here's a quick test to find out how your machines behave. On a vertical machining center, give this command in the Imperial mode:

G91 G20 G01 X1.0 Y0.001 F5.0

When you run this command, monitor the position display. You'll likely see the machine move in X (only) for about 0.1 inch – then the Y axis will step by 0.0001. This will be repeated nine times. Now give the equivalent command in the Metric mode:

G91 G21 G01 X25.4.0 Y0.025 F125.0.

You'll still see the machine make steps, but they will be much closer together – (0.1 mm, or about 0.04 inch apart). The tool path is being kept closer to its programmed path.

What happens when you switch measurement systems?

CNC machines are initialized to start up in the user's measurement system mode of choice. A parameter controls which mode. Additionally, two G codes can be used to specify measurement system mode (commonly G20 for the Imperial system and G21 for the Metric system).

Most CNC people agree that it isn't really feasible to incorporate both measurement systems modes within the same program (though I'd bet there are people doing it). Most programmers will run a job entirely in their measurement system of choice. But there are many companies that will switch measurement system modes quite regularly, based upon how a given workpiece drawing is dimensioned.

When the measurement system is changed, it is important to know what happens to all values within the CNC control (axis displays, offsets, parameters, etc.). With some – especially older – machines, the decimal point will simply move one place to the right or left. With these machines, when switching from Imperial to Metric, a value of 10.0000 inches will be changed to 100.000 mm. 100.0 mm, of course, is not equivalent to 10.0 inches, so this makes switching measurement systems quite difficult (every offset value must be changed).

Also, remember that certain program-related parameters are affected by the current measurement system mode. If you use the G83 peck drilling cycle, for example, the clearance distance during each peck is controlled by a parameter. If the machine doesn't truly convert values during a measurement system change, these parameters must also be changed. This makes changing measurement systems all that more difficult.

With newer machines, a true conversion takes place. A value of 10.0000 inches will be shown as 25.400 mm when switching from Imperial mode to Metric mode. This eliminates the need to change any values that will be needed after the measurement system mode is changed.

Note that a parameter setting might control what happens in this regard. So if you notice that the conversion does not take place when switching modes – especially with newer machines – be sure to search out the parameter that controls this function.

Making programs for similar machines compatible

Many CNC users have two or more highly similar CNC machines made by different machine tool builders. Depending on work load, most need to run the same jobs on any one of the similar machines at any given time. Say for example, you have two twenty horsepower, eight inch chuck, turning centers. One is made by brand X and the other by brand Y. As each job comes up you would probably like to run it on which ever turning center frees up first.

One problem this kind of CNC user constantly faces is incompatibility with regard to programs. Even if the two (or more) similar machines have identical controls, there will probably still be some incompatibility regarding M codes.

If both of the turning centers have two spindle ranges, one may use M23 for low range and M25 for high range. The other may use M41 and M42. To overcome these M code incompatibility problems, either the CNC user must maintain two (or more) complete sets of CNC programs (which can be cumbersome), or the operator of each machine will be constantly editing programs written for the other machine (which is tedious, error prone, and time consuming).

Turning center spindle ranges are but one example of when similar CNC machines built by different machine tool builders are not compatible with regard to M codes. While some M codes are somewhat standardized (M00, M01, M03, M04, M05, etc.), almost any M code the machine tool builder must name for their own miscellaneous functions introduces a potential incompatibility problem (tailstock, chuck jaws open/close, automatic tool changer functions, pallet changer, etc.).

If your machines have custom macro B, you have the ability to change the function of your M codes (up to ten on current model controls) in such a way that the machines become compatible. In

essence, you can set up the control so that when it reads one M code, it executes another. While this technique can be applied to any M code incompatibility, let's stress its use for the turning center spindle range M codes.

The first step is to pick the M codes you want to use. In the spindle range example, maybe more of your machines use M41 and M42, so you wish to change the machines that currently use M23 and M25. Or maybe the machines that use

M41 and M42 do not have custom macro B, meaning the only machines you can change are those that use M23 and M25 (though custom macro B can be added to many controls at any time).

For those machines you wish to change (we'll say those using M23 and M25), look in the custom macro section of your programming manual. In the section related to creating your own M codes, you will find parameters that must be changed (two in our case). These parameters control what happens when a specified M code is executed. When set properly, whenever the specified M code is read, the control will automatically jump to a custom macro. In the custom macro, we will include the M code that performs the actual spindle range change. This is a bit confusing, so here is a full example.

One popular control uses parameter numbers 7071 through 7079 to control to control which M codes call custom macros. For this control, parameter number 7071 is related to program number O9001 and parameter number 7072 is related to program number O9002. On the machines that currently use M23 and M25, we will set parameter number 7071 to a value of 41 and 7072 to a value of 42. From this point on, whenever this control reads an M41, it will execute program number O9001. Whenever it reads an M42, it will execute program number O9002.

The last step is to enter the correct programs into the control. Most controls even allow your M code related custom macros to be protected so that they cannot be changed or deleted (another parameter is related to this function). Here are the two simple programs.

- O9001 (Program number)
- M23 (Execute low range command)
- M99 (End of custom macro)

- O9002 (Program number)
- M25 (Execute high range command)
- M99 (End of custom macro)

When these programs are in the control's memory, and when the parameters are properly set (remember, they will probably be different parameter numbers for your particular control), whenever the control reads an M41, it will jump to program number O9001. In this program, the M23 will make the range change. From this point on, programs that have M41 and M42 will execute without modification in machines originally set up to accept M23 and M25!

You say your turning center has only one spindle range?

There may be times when you find the need to delete M codes in order to get programs to run. In the previous example, maybe one of the similar turning centers has only one spindle range. Yet whenever an M41 or an M42 is read,

the control of this machine will stop executing due to an unrecognizable M code. For this type of machine, you can use similar techniques to make the control ignore the unused M codes. The parameters will be set the same as shown earlier. Only the custom macros will change.

- O9001 (Program number)
- M99 (End of program)

- O9002 (Program number)
- M99 (End of program)

Notice these are nothing more than empty programs. When the control reads an M41, it will jump to O9001, which immediately sends execution back to the previous program. Though nothing actually happens within the programs, the control will no longer get stuck on the unrecognizable M codes!

Determining which machine is being run from within CNC programs

It is sometimes necessary to run a given job on several (possibly different) CNC machine tools. For instance, the required number of workpieces may be greater than can be produced on a single machine. Or maybe you have multiple machines with similar attributes, and a given job may be run on whichever machine becomes available first.

When multiple (and especially different) CNC machines are involved with running a single job, it will normally require multiple CNC programs. This is because there may be machine-specific commands within each program, like M-code usage and tool change position. Creating, storing, and maintaining a separate program for each machine is cumbersome. And if a design engineering or process change is required, all of the related programs will have to be changed.

This is a seamless way to ensure that one program could be run among different machines. Our example application is for ensuring that work coordinate system offsets are appropriately set, but this technique can be helpful for many applications. I must point out one important limitation: it requires similar CNC controls that allow access to parameter settings. The example we provide is for FANUC CNCs.

The first step is to choose a CNC parameter that is available on all of the related machines but that has a different value in each. Once this parameter is determined, make note of its value. We chose a parameter related to axis stroke limits, and our example will do the same. After confirming that this parameter is set differently on each machine, the second step is to test against this parameter setting

(from within the program) whenever a machine-specific programming command is required. The third and final step is to specify the series of commands that vary from one machine to the other.

Here is a specific example. Say that for one machine (we'll call it machine A), standard fixture offsets are used (G54-G59). But the other machine has the optional fixture offset option (G54.1 Pxx). Our universal program must be able to specify the appropriate fixture offset invoking word/s code regardless of which machine is currently running it.

We'll use the X axis stroke limit to determine which machine is currently being used. The X axis stroke limit for a current model FANUC CNC is stored in register number one (the X register) of parameter 1321. The FANUC function that provides access to parameter settings is PRM. Its syntax for this function when accessing parameters that have multiple registers is:

- PRM[*num*] / [*reg*]

The label *num* represents the parameter number and the label *reg* represents the register number. Here is a command that will access the X register (register number 1) for parameter number 1321 and store the result in common variable #100.

- #100 = PRM[1321] / [1]

If you elect to use this parameter for testing, of course, you must first make note of its value on each machine, and make sure that there is a different value in this parameter for each machine. In machine A, say we find the X register of parameter 1321 to be a value of 15.375. In machine b, this value is 17.436. Sure enough they are different, so we make note of them.

Now a simple test can be made using a conditional branching command (IF statement) to determine which machine is being used. Here is a sample program that shows how.

- O0001 (Program number)
- G91 G28 Z0 M19
- T01 M06
- G90 G54 S1500 M03
- G00 X1.0 Y1.0
- IF [PRM[1321]/[1] EQ 15.375] GOTO 1 (Machine A is being used)
- IF [PRM[1321]/[1] EQ 17.436] GOTO 2 (Machine B is being used)
- #3000 = 100 (MACHINE IS NOT RECOGNIZED)
- N1 G54
- GOTO 3
- N2 G54.1 P1
- N3… (Program continues)
- .
- .
- .

The point of all of this is that you can make your program access unique parameter settings to discover which machine is being used. This has tremendous implications for unifying programs, regardless of application complexity.

Maintain consistency in program structure

There are a lot of repeated commands in CNC programs. Some are seemingly redundant – but necessary. And many can be grouped into five groups. Here is an outline showing the groups:

- Program startup commands
    - Commands to machine with the cutting tool
- Tool startup commands
    - Commands to machine with the current cutting tool
- Tool ending commands
- Tool startup commands
    - Commands to machine with the current cutting tool
- Tool ending commands
- Tool startup commands
    - Commands to machine with the current cutting tool
- Tool ending commands
- Program ending commands

Ideally, the commands for every cutting tool should be independent of the rest of the program. This is especially important for machines that allow operators to rerun tool, like machining centers and fixed headstock turning centers. This means certain word may seem redundant, but they are necessary if a cutting tool is to be rerun.

Keep your program structure “clean” and consistent from program to program. Computer aided manufacturing (CAM) systems are notorious for generating G-code in a rather haphazard manner. While programs may run correctly, this can be very confusing to the CNC operator.

Here is an example of a well-structured program for a CNC vertical machining center:

Program-startup structure

```
%
O0026 (EXAMPLE PROGRAM FORMAT)
N010 G17 G20 G94 (SELECT XY-PLANE, INCH MODE, SELECT FEED-PER-MINUTE MODE)
N020 G40 G64 (CANCEL CUTTER COMP. AND SELECT NORMAL CUT. MODE)
N030 G69 G80 (CANCEL ROTATION AND CANNED CYCLES)
```

Tool-startup structure

```
(0.25-IN DRILL)
N100 T01 M06 (LOAD TOOL IN SPINDLE)
N110 G54 G90 S1200 M03 T02 (START SPINDLE, READY NEXT TOOL)
N120 G00 X1.0 Y1.0 (MOVE TO XY APPROACH POSITION)
N130 G43 H01 Z0.1 (INSTALL TOOL LEN. COMP., MOVE TO Z APPROACH PSN)
N140 M08 (COOLANT ON)
N150 G01 Z-0.65 F4.0 (CUTTING MOVES WITH FEEDRATE)
N160 G00 Z0.1 (RETRACT FROM HOLE)
```

Tool-end structure

```
N170 M09 (COOLANT OFF)
N180 G91 G28 Z0 M19 (TOOL CHG. PSN., PRE-ORIENT SPINDLE)
N190 M01 (OPTIONAL STOP)
```

Tool-startup structure

```
(0.375-IN DRILL)
N200 T02 M06 (LOAD TOOL IN SPINDLE)
N210 G54 G90 S2000 M03 T03 (START SPINDLE, READY NEXT TOOL)
N220 G00 X2.0 Y1.0 (MOVE TO XY APPROACH POSITION)
N230 G43 H02 Z0.1 (INSTALL TOOL LEN. COMP., MOVE TO Z APPROACH PSN)
N240 M08 (COOLANT ON)
N250 G01 Z-0.7 F5.0 (CUTTING MOVES WITH FEEDRATE)
N260 G00 Z0.1 (RETRACT FROM HOLE)
```

Tool-end structure

```
N270 M09 (COOLANT OFF)
N280 G91 G28 Z0 M19 (TOOL CHG. PSN., PRE-ORIENT SPINDLE)
N290 M01 (OPTIONAL STOP)
```

Tool-startup structure

```
(0.5-IN DRILL)
N300 T03 M06 (LOAD TOOL IN SPINDLE)
N310 G54 G90 S800 M03 T01 (START SPINDLE, READY NEXT TOOL)
N320 G00 X3.0 Y1.0 (MOVE TO XY APPROACH POSITION)
N330 G43 H03 Z0.1 (INSTALL TOOL LEN. COMP., MOVE TO Z APPROACH PSN)
N340 M08 (COOLANT ON)
N350 G01 Z-0.75 F6.0 (CUTTING MOVES WITH FEEDRATE)
N360 G00 Z0.1 (RETRACT FROM HOLE)
```

Tool-end structure

```
N370 M09 (COOLANT OFF)
N380 G91 G28 Z0 M19 (TOOL CHG. PSN., PRE-ORIENT SPINDLE)
N390 M01 (OPTIONAL STOP)
```

Program-end structure

```
N400 G28 X0 Y0 (RETURN TO XY ZERO RETURN – LOAD/UNLOAD PSN)
N410 M30 (END OF PROGRAM)
%
```

This program simple drills three holes, so there are not many machining commands. Note the repeated structure. Lines N100 through N140 have the same structure as lines N200 through N240 – and lines N300-N340.

Note the documentation. While the documentation in this program may be overkill (every line is documented), the essential documentation in this program would be the tool names at the beginning of each tool.

Also note the sequence numbers. While they are optional, they help an operator know that they are at the right place in the program. This is especially helpful with longer programs.

And note the skipping of lines between tools. This is done by specifying an end-of-block character in a line by itself. This clarifies where each new tool begins. If done consistently, the operator will be able to easily identify the restart command for each tool (lines N100, N200, and N300).

Obviously, the words in the program structure for your machines will be different from what we show in the example. But you can apply this logic to any form of multi-tool CNC machine.

One more point. Note the bolded words (Y1.0). This is an example of what you might consider to be a redundant word. Once the Y-axis is moved into position in line N120, the Y-axis does not move for the rest of the tools in the program. Some programmers may choose to leave out the Y1.0 words in lines N220 and N320. But this would eliminate the ability to rerun tools two or three (by themselves). If the entire program is run, notice that the Y-axis will be left at its reference position at the end of the program (line N400). The Y1.0 words are mandatory in lines N220 and N230 if you expect operators to be able to rerun these tools.

Maintain consistency among programs: Cutter radius compensation variations

Cutter radius compensation for machining centers (which may also be called cutter *diameter* compensation) has at least three major benefits. It lets a manual programmer ignore cutter size when calculating coordinates for the program, it allows the setup person to choose from a range of cutter sizes, and it allows for easy trial machining and workpiece sizing during setup and the subsequent production run. Programmers must fully understand and master this important CNC feature.

There are actually two methods of applying cutter radius compensation. First, let's describe the two methods. Then we'll make a few comments about which method is best for a given company.

Method one: programmed path is work surface and offset is the cutter's size

With method one, the programmed path is a series of coordinates that are right on the workpiece. These coordinates are quite easy to determine, many times coming right from the blueprint. The setup person will specify the size of the cutter (its radius or diameter, depending upon the control manufacturer) in the cutter radius compensation offset. When the program is run, the control will keep the cutter away from the programmed path (the work surface) based upon what it sees in the offset, effectively machining the contour to its desired size.

For example, if the setup person uses a 1.0 inch cutter, and if the control manufacturer requires that the offset be specified in radius (not diameter), the cutter radius compensation offset will be set to 0.5 inch.

Method two: programmed path is cutter centerline path and offset is the deviation from planned cutter size.

With method two, the programmed path is based upon a *planned cutter size*. The programmed coordinates will be the *centerline path* of this cutter as it machines the contour. In the offset, the setup person will specify the deviation from the planned cutter size to the size of cutter actually being used. Again, control manufacturers vary when it comes to whether the offset must be specified as radius or diameter deviation.

For example, if the programmer plans for a 1.0 inch cutter, all programmed coordinates will be kept 0.5 inch from the work surface. If the setup person uses a 1.0 inch cutter, the cutter radius compensation offset will be zero (the planned cutter size is being used). If the setup person uses a 0.75 diameter cutter, the actual cutter size is smaller than the planned cutter size, and the offset value will be set to -0.125 (assuming the control requires the deviation to be specified in radius). If the setup person uses a 1.25 diameter cutter, the offset will be set to 0.125 (the actual cutter size is larger than the planned cutter size).

Which method is better?

Because work surface coordinates are so much easier to calculate, most manual programmers prefer method one. The more complicated the contour to be programmed, the more manual programmers cling to method one. A computer aided manufacturing (CAM) system can calculate centerline path coordinates just as easily as it can calculate work surface path coordinates. This means CAM system programmers tend to have more of a choice to make.

Your choice should be base upon simplifying the use of the CNC machine for setup people and operators. Remember, they're the people that actually make your company's money. Keep it as simple as possible for them. So the question becomes, "Which method minimizes the potential for mistakes at the machine?" Just choose the method that best answers this question for your own company.

Though this may open the door to a great deal of debate, here comes my opinion. People tend to be too quick to jump to the conclusion that method one is best. Admittedly, the cutter's initial radius (or diameter) should be quite easy to determine. For new cutters, just look at the cutter. It's size is etched on the shank. For re-sharpened cutters, measure it with a micrometer. If your control requires a radius value, divide the cutter diameter by two.

With method two, the setup person must know the intended cutter size. It should be included in the setup documentation and a message should be placed at the beginning of the CNC program so it appears on the control's display screen. If your company uses a great percentage of new cutters, the related offset values will be zero, meaning the setup person may have nothing to do, eliminating the need for any kind of action.

While the initial offset entry is very important, remember that at least it is done by a highly skilled setup person. If the production run is to be turned over to a lesser-skilled CNC operator, and if sizing due to tool wear must be done during the production run, you must also consider any difficulties the operator will have with cutter radius compensation. If you use method one, the operator will be dealing with a rather large value in the cutter radius compensation offset. While most current controls allow operators to incrementally change offset values by small amounts, if they make a mistake, it will not be very obvious. If the initial value of the offset is close to zero, as is commonly the case when using method two, an entry mistake will stand out.

Be consistent!

Again, this choice must be based upon simplifying things for setup people and operators. Judge the wisdom of your current choice by how many scrap-causing mistakes are currently being made. But whichever method you choose, stick with it for all programs.

When two or more programmers are involved – or maybe different programmers have programmed a given machine over the years, there may be some programs that specify cutter centerline path coordinates and others that specify work surface path. This dramatically complicates the specification of cutter radius compensation offset values, and can lead mistakes that cause scrap parts – or worse.

Maintain consistency among programs: Tool length compensation variations

As with cutter radius compensation, there are two poplar methods for using tool length compensation:

1. Method one: Offset value is the length of the tool and the Z-axis program zero assignment (fixture offset) value is the distance from the spindle nose with the Z-axis at its reference position to the Z-axis program zero point.
2. Method tow Offset value is the distance from the tool tip to the Z-axis program zero point and the Z-axis program zero assignment (fixture offset) value is set to zero.

Again, what is most important is that you choose a method and stick to it for all programs on every machine in your shop. Frankly speaking, this is what most companies do.

We recommend that you use method one for three reasons:

1. Tool offset values can be measured off line, possibly by someone in the tool crib.
2. Offsets will remain the same for tools used from one job to the next.
3. Tools (and offsets) can be shared from one machine to another without having to re-determine the offset value.

That said, method two is quite popular when one person is responsible for all setup- and production run-related tasks, as is the case in many workpiece producing companies (job shops).

Ease of use

We now switch the focus to making programs as operator-friendly as possible. It should go without saying that there are countless ways to help operators when running programs. Anything time you

see an operator struggling with a program, it should be taken that there is something you could do to help them.

Documenting in the program

With FANUC and FANUC-compatible CNCs, parentheses [()] are used to enclose documenting messages. We offer several times when you should include them to help operators in one way or another.

Program headers

It is amazing how many programs I have seen, even recently, where the line right after the program number (or name) begins giving CNC commands. Every CNC program should begin with some documenting comments that say something about the program: what it is, what it is used for, who wrote it – and when, among other things.

Here is an example showing the kinds of things you should include about the program. You can probably think of other things that would help in your own CNC environment.

- %
- O0001
- (*** PROGRAM QUALIFIED 2/12/99, ML ***)
- (MACHINE: MORI SEIKE SL4)
- (PART NUMBER: A-2355-2C)
- (PART NAME: BEARING FLANGE)
- (REVISION: F)
- (CUSTOMER: ABC COMPANY)
- (OPERATION: 20, MACHINE BORED END)
- (PROGRAMMER: MLL)
- (DATE FIRST RUN: 4/11/16)
- (PROGRAM REVISION: C)
- (LAST PROGRAM REVISION: 1/30/20 BY CRD)
- (RUN TIME: 00:05:25)
- N005 T0101 M41
- N010 G96 S400 M03
- N015 G00 X3. Z.1 M08
- .

Though most of the comments are self-explanatory, let me elaborate a bit.

Many companies strive to “qualify” their programs, meaning they eliminate the possibility that anything can change from one time the program is run to the next. Qualifying a program may require several runnings of the job, but once a program is qualified, it should flawlessly run without any issues cropping up. The first documenting message specifies that this program is qualified, when that happened, and who deemed it so.

One very important message specifies the workpiece revision. There may be many versions of the program floating around based upon the various revisions the workpiece has gone through. Without this piece of information, the operator could easily run the wrong version of the program.

The last item I will mention is run time. Once the program has successfully completed a production run, including this in the program header will allow anyone looking at the program (even when the job is not running) will know how long the program takes to run. This could be helpful for planning purposes, and will help an operator check that everything is as it should be each time the job is run. A dramatic difference in current run time may indicate that something is wrong.

Tool/process names and current size specification

Help operators identify the commands related to cutting tools. Each tool should begin with a message identifying the cutting tool and/ or the process being performed. It should end with a message specifying what the cutting tool has done – in a command just after the optional stop command (M01 for each tool. Consider the bolded commands in the program below.

- %
- O0002 (Program number)
- N005 G20 G18 G99
- N010 G50 S4000
-
- **(ROUGH FACE AND TURN)**
- N005 T0101 M41
- N010 G20 G96 S500 M03
- N015 G00 X3.2 Z0.005 M08
- N020 G99 G01 X-0.062 F0.012
- N025 G00 Z0.1
- N030 X2.33
- N035 G01 Z0
- N040 X2.58 Z-0.12
- N045 Z-1.495
- N050 X3.2
- N055 G00 X8.0 Z7.0
- N060 M01 (Optional stop)
- **(DIAMETER SHOULD BE 2.580)**
-
- **(DRILL CENTER HOLE)**
- N065 T0202 M41
- N070 G97 S1150 M03
- N075 G00 X0 Z0.1 M08
- N080 G01 Z-2.705 F0.008
- N085 G00 Z0.1
- N090 X8.0 Z7.0
- N095 M01
- **(1" HOLE MUST BE 2.405 DEEP)**
-
- (ROUGH BORE)
- N100 T0303 M41
- N103 G96 S400 M03
- N105 G00 X1.585 Z0.1 M08
- N110 G01 Z0 F0.007
- N115 X1.46 Z-0.0575
- N120 Z-1.87
- N125 X0.95
- N130 G00 Z0.1
- N135 X8.0 Z7.0
- N140 M01
- **(HOLE DIAMETER MUST BE 1.460)**
-
- **(FINISH BORE)**
- N145 T0404 M42
- N150 G96 S500 M03
- N155 G00 X1.625 Z0.1 M08
- N160 G01 Z0 F0.005
- N165 X1.5 Z-0.0625
- N170 Z-1.875
- N175 X0.95
- N180 G00 Z0.1
- N185 X8.0 Z7.0
- N190 M01

- **(HOLE DIAMETER MUST BE 1.500)**
-
- **(FINISH FACE AND TURN)**
- N195 T0505 M42
- N200 G96 S500 M03
- N205 G00 X2.7 Z0 M08
- N210 G01 X1.3 F0.005
- N215 G00 Z0.1
- N220 X2.25
- N225 G01 Z0
- N230 X2.5 Z-0.125
- N235 Z-1.5
- N240 X3.2
- N245 G00 X8.0 Z7.0
- N250 M01
- **(OUTSIDE DIAMETER MUST BE 2.500)**
-
- N250 M30 (End of program)
- %

The first message in each tool clearly identifies the process being performed – and the operator will easily know that the next command begins the tool (and is its restart command.

The message at the end of each tool will help the setup person size in each tool when running the first workpiece. This will be especially helpful for machines that allow cutting tools to be rerun (like machining centers and fixed headstock turning centers). As long as the optional stop switch is turned on, the machine will stop at the end of each tool. The setup person will see the related message, telling them what the tool has done and/or what size the machine surface must be. They will then measure the surface, and if necessary, make the sizing adjustment and possibly rerun the tool.

When doing something out of the ordinary in the program

We have been stressing the importance of consistency as it relates to helping operators get familiar with and keep abreast of thing that happen in a CNC program. But there are times when you need to do something in a program that you do not normally do. Whenever this happens, be sure to make it clear with documenting messages in the program. Consider this example

- %
- O0002 (Program number)
- (****************** SPECIAL NOTE ******************)
- (THE GROOVING TOOL IN STATIO #5 USES TWO OFFSETS.)
- (OFFSET #5 CONTROLS THE GROOVE IN THE 1.375 INCH DIA.)
  (OFFSET #25 CONTROLS THE GROOVE IN THE 4.25 INCH DIA.)
-
- N005 G20 G18 G99
- N010 G50 S4000
-
- (ROUGH FACE AND TURN)
- N005 T0101 M41
- N010 G20 G96 S500 M03
- N015 G00 X3.2 Z0.005 M08
- N020 G99 G01 X-0.062 F0.012
- N025 G00 Z0.1
- .

- .
- .

When making program changes

Most companies allow CNC programs to be changed in a rather haphazard manner. Changes are often made "on the fly" after some issue crops up that causes a problem with the running of the program – and no one documents the issue or the fact that the program was changed. This can lead to confusion down the road when the program is run again and the issue causing the change is no longer a factor.

Additionally sometimes you may be asked to make program changes that you do not agree with. Maybe you are asked to increase cutting conditions and you are worried this will cause problems with tool life. If the problem does eventually occur, you may be the person held responsible since you did make the change.

Documenting in the program will help in either case. Indeed, it would be wise to include a program message any time the program is changed. The time saved by not documenting will be quickly lost if the change eventually causes confusion in the future. Consider this example:

- N100 T0303 M41
- N103 G96 S600 M03 (SPEED INCREASED FOR EFFICIENCY 3/20/21 PER WC)
- N105 G00 X1.585 Z0.1 M08
- N110 G01 Z0 F0.015 (FEEDRATE INCREASED FOR EFFICIENCY 3/20/21 PER WC)
- N115 X1.46 Z-0.0575
- N120 Z-1.87
- N125 X0.95
- N130 G00 Z0.1

Anyone looking at this program in the future will know what was changed, why, and who requested the change.

Simplify setup-related tasks: Program zero assignment

There are things you can do in your programs to help setup people get a job ready to run production. This help can have a big impact on productivity – so always watch for time when they struggle. Consider ways CNC programs can help. We begin with techniques that can help with program zero assignment.

Eliminate Program Zero Assignment Tasks with Qualified Setups

A *qualified* setup is one that can be accurately repeated over and over again. When it comes to machining center setups, a qualified setup can ensure that the workpiece/s (and specifically the program zero point location/s will be in exactly the same place ever time the setup is made. If the program zero point location is exactly the same time after time, all tasks related to program zero assignment can be eliminated, which of course, will reduce setup time.

While many companies go to great lengths to ensure that their setups are qualified. Generally speaking, this involves making fixtures that are keyed to the machining center table in some way (to key slots, in dowel holes, etc.). But few reap the total benefit of qualifying their setups. Their setup people still have to perform certain tasks during setup relative to the assignment of program zero.

Eliminating the measurements required to assign program zero with qualified setups

If a setup is truly qualified, there is no reason to measure the program zero locations during setup. The same program zero assignment values used the last time the setup was made can be used the next time the setup is made. At worst, if the programmer cannot *predict* the program zero assignment values as the program is written (possibly due to a fixture not being made perfectly to print), the program zero setting values need only be measured the very first time the setup is made.

Some setup people have trouble believing that setups are truly qualified. They just cannot be convinces that a fixture can be replaced accurately enough. For this reason, they measure program zero in every setup, wasting a great deal of setup time. If you have this problem, at the very least, program the machine to go to each program zero position (a special series of commands can be included in your machining program). An M00 program stop can then be given so the setup person can simply *confirm* that the program zero point location is correct. After finding that it is time after time, your setup people will eventually start trusting that your setups are truly qualified.

What about crashes?

One reason I constantly hear against trusting that setups are truly qualified has to do with crashes. After a crash, some machines cannot be perfectly re-aligned (though most can with proper procedures). If an axis is not perfectly re-aligned, all program zero assignments made to this point will no longer be correct. But remember that most controls have a feature that allows you to shift the machine's coordinate system with one simple fixture offset entry. FANUC calls this the common fixture offset. After a crash, and assuming the machine cannot be perfectly re-aligned, you must measure (one time) to find out how much misalignment exists and enter the misalignment values into the common fixture offset.

Eliminating the fixture offset entry

If setups are qualified, not only can you eliminate the program zero assignment value measurements, you can also eliminate the actual *entry* of the corresponding fixture offset values. Almost all current model CNC controls let you *program* fixture offset value entries. One popular command for this purpose is G10. Here is an example of the G10 command for one popular control model.

- G90 G10 L2 P1 X-12.3736 Y-10.2376 Z-8.3847

The L2 in this command specified that fixture offsets are being set (as opposed to other offset types). The P one specifies the fixture offset number. And the X, Y, and Z values are the values that are being entered into the fixture offset. Note that this command can be included in the CNC program and will set the values of fixture offset number one.

Facilitate the measurements required to assign program zero with a spindle probe or edge finder

If you cannot justify what it takes to qualify setups, of course, you cannot eliminate the task of program zero assignment. You can, however, help the setup person measure and enter program zero assignment values.

The best way is to use a spindle probe. This device pretty much automates the process. The setup person simple positions the probe stylus (manually) to bring it to a pre-planned location relative to the program zero surface and activates the appropriate probing program. The probe does the rest, measuring each program zero assignment value and entering it into the appropriate fixture offset register.

But not every machine has a spindle probe. The next best thing is to use an edge finder as if it were a spindle probe. I recommend using a conductivity-type edge finder that lights up the instant it touches something. The only real difference from a spindle probe is that the setup person will have to manually touch each program zero surface.

Here is how the technique works for the corner finding function. The operator will first manually position the edge finder to within about 0.5 inch of the corner to be picked up and activate the corner finding custom macro program. The program will automatically move the edge finder into a location at which one surface can be picked up (say the X surface) and stop with M00. The operator will then place the machine in the handwheel mode and pick up the left surface. When finished, they will then place the machine back in automatic mode and reactivate the cycle. The program will continue by calculating the program zero point value in X and automatically place this value into the corresponding fixture offset. These techniques will be repeated for the other two surfaces.

Instead of using the skip cutting system variables (#5061, #5062, & #5063) as are needed when probing to come up with the machine position after touching a surface, we will instead use system variables that constantly contain the distance from the reference position to the machine's current position (#5021 for X, #5022 for Y, and #5023 for Z). Also, our example will assume the edge finder radius is stored in permanent common variable #500 and its length is in #501.

- O9051 (Corner pickup routine for lower left corner)
- N1 G91 G01 Y0.75 Z-0.75 F30. (Move to first touch position)
- N2 #3006 = 101 (TOUCH LEFT SIDE IN X)
- N3 G90 G10 L2 P1 X[#5021 + #500] (Set fixture offset X)
- N4 G91 G01 X-0.2 (Move away in X)
- N5 Y-0.75 (Move down in Y)
- N6 X0.55 (Move to second touch position)
- N7 #3006 = 101 (TOUCH BOTTOM SURFACE IN Y)
- N8 G90 G10 L2 P1 Y[#5022 + #500] Set fixture offset Y)
- N9 G91 G01 Y-0.2 (Move away in Y)
- N10 Z0.75 (Move up in Z)
- N11 Y0.55 (Move to third touch position)
- N12 #3006 = 101 (TOUCH TOP SURFACE IN Z)

- N13 G90 G10 L2 P1 Z[#5023 - #501] (Set fixture offset Z)
- N14 G91 G01 Z.5 (Move away in Z)
- N15 X-0.75 Y-0.75 (Move away in X and Y)
- N16 M30 (End of program)

At each M00, the operator will place the machine in the handwheel mode and pick up the requested surface. Then they will place the machine back in the automatic mode and reactivate the program (by simply pressing cycle start). At the completion of this program all three values in fixture offset number one will be set (also eliminating the possibility for operator entry errors).

Centering the Y axis for CNC lathes

More and more CNC lathes have live tooling capabilities – especially sliding headstock machines. This allows them to perform the same kinds of machining operations done on milling machines, eliminating the need for secondary operations. Rotating cutting tools, like drills, taps, reamers and end mills can be held in at least two attitudes: parallel to the X axis or parallel to the Z axis. These cutting tools can perform machining operations on faces (Z axis work) or on diameters (X axis work).

With some live tooling machines, each cutting tool is fixed on the X and Z axis centerlines. That is, the tool cannot move in the direction perpendicular to the XZ plane. These machines can only perform machining at the center of the workpiece in X.

More sophisticated live tooling machines *do allow* machining perpendicular to the XZ plane. They have a Y axis (though not all machine tool builders call this axis the Y axis). While the Y axis often has a rather limited amount of travel (compared to the Y axis of a machining center), this dramatically increases the capability of the machine. Now the machine can perform machining operations that are not centered on the workpiece.

One common frustration with Y axis lathes is that the each cutting tool requires *calibration* during setup. This is because, with many machines, live tooling tool stations are not perfectly aligned with one another. When the center of one of them is at the X axis center, the others won't be. Indeed, many sliding headstock machines actually use the Y axis as the "tool changing mechanism". The Y axis simply moves to change from one cutting tool to the next.

While there may be ways to permanently qualify and record the related center positions, many setup people go through a rather tedious process to calibrate every live tool along the Y axis for every setup. This procedure involves touching off an aligning pin (held in the tool station) on both sides of the workpiece and using the position displays to calculate the center of workpiece location for the tool station along the Y axis. With this value known (the Y axis program zero assignment value for the tool station), the setup person will enter it into the appropriate geometry offset.

This program will dramatically simplify the process:

- O0001 (CROSS TOOL ALIGN Y AXIS)
- N1 #121 = 1 (TOOL STATION NUMBER TO CALIBRATE)
-
- (INDEX TURRET AND JOG PIN TO ABOUT 0.25 ABOVE PART IN X)

-
- N2 G98
- N3 #101 = #5003
- N4 G01 Y[#101+0.4] F30.0
- N5 U-1.0
- N6 #3006 = 100 (TOUCH X PLUS SIDE)
- N7 #110 = #5003
- N8 G01 Y[#101+0.4]
- N9 U1.0
- N10 Y[#101-0.4]
- N11 U-1.0
- N12 #3006 = 100 (TOUCH X MINUS SIDE)
- N13 #111=#5003
- N14 Y[#101-0.4]
- N15 U1.0
- N16 #105 = [[#110+#111]/2]]
- N17 Y#105
- N18 #106 = #5023
- N19 #[2300 + #121] = #106
- N20 G99
- N21 M30

The setup person will begin by placing a pin in the cutting tool holder (as they are probably doing currently). The pin can be the same diameter as the cutting tool they'll be using.

They will then alter line N1, entering the tool station number of the tool to be calibrated. Next, they manually jog the machine so that the pin is approximately centered on the workpiece in Y and about 0.2 inch above the workpiece in X.

Next, they will activate this program. In line N3, the current Y position is stored. #5003 is the current position in the *third axis*, which for most turning centers is the Y axis. (You must confirm this for your machine before using the custom macro.)

The Y axis will move plus in line N4 to a position that will clear the workpiece diameter. In line N5, the pin will move below center in X (U commands an incremental X movement).

In line N6, the machine will stop and place the message "TOUCH X PLUS SIDE" on the display screen. The setup person will select the handwheel mode and cautiously touch to pin to the workpiece in X. When finished, they will place the mode switch back to automatic mode and press the cycle start button.

Line N7 stores the current Y position (while the pin is touching) and then the pin will move clear and come back to center in lines N8 through N10.

The process is repeated for the Y minus side in lines N11 through N15. In line N16, the true Y centerline is calculated, and the Y axis moves to this position. In line N18, the current Y axis position relative to the Y axis zero return position is determined. Finally, in line N19, this value (Y axis program zero assignment value) is placed in the appropriate geometry offset.

This program can be improved. You may, for example, be able to come up with a way to have the custom macro automatically detect which live tool is in position (possibly by testing the current Y position after center is found). Also, this program is for relatively small parts (up to about 0.5 inch in diameter if a 0.25 diameter pin is used). With a few modifications, it can be made to work for larger part sizes.

Simplify setup-related tasks: Cutting tools

Here are some things you can do with programs to simplify certain tasks related to setting up cutting tools.

Facilitate tool offset entries with a program created by tool length measuring device

This technique only applies if you have someone (in the tool crib) assembling and measuring cutting tools for machining centers. Commonly, the tool length values for each tool and cutter radius values for side milling cutters are written down and passed along with the cutting tools. The operator loads them into the tool changer magazine and enters the offset values the tool crib attendant has written down.

If this describes your methodology, you may want to update your thinking. Current tool length measuring devices can output the tool data to a program with G10 commands. The program is loaded into the CNC and run once to enter all tool data. This eliminates the potential for entry errors (by the tool crib attendant and the CNC operator) and saves time as long as you have an efficient distributive numerical control (DNC) system. The more tools you have in your jobs, the bigger the benefits.

Here is an example of a program generated by the tool length measuring device:

- %
- O8001 (Offset program)
- (GENERATED 6/13/19, 2:34pm)
  (JOB NAME: REAR SPINDLE, FLANGE END, OP 10)
- (MACHINE: NIGATA MC30)
- G90 G10 P1 R6.5893
- G10 P2 R6.3321
- G10 P3 R5.4678
- G10 P4 R7.5746
- G10 P5 R3.4858
- G10 P6 R4.3433
- G10 P7 R8.4872
- G10 P8 R5.3876
- G10 P9 R8.9383

- G10 P10 R20.0000 (MUST MEAURE AT THE MAHINE!)
- G10 P11 R6.5893
- G10 P12 R5.3321
- G10 P13 R7.4678
- G10 P14 R5.5746
- G10 P15 R3.4858
- G10 P16 R5.3433
- G10 P17 R4.4872
- G10 P18 R6.3876
- G10 P19 R5.9383
- M30
- %

A couple of comments about the program. First note the messages that specify what these offsets are for, including the date and time the tools were measured. The operator can easily ensure the correct program is being used.

Also note the 20.0000 value for tool ten (specified with R10). There may be times when the tool setter cannot assemble or measure a tool for some reason. Maybe the components were not yet available at the time of measuring. In this case, the value going into the offset is bigger (longer) than the longest tool the machine can hold. If the operator happens to miss this and runs the program, the worst that could happen is the machine will overtravel when making its approach movement with this tool.

Facilitate tool length measuring on the machine with a tool length measuring probe or a program

If your machine does not have a tool length measuring probe, you can use the same features used for program zero assignment with an edge finder to help an operator measure tool lengths on the machine. Tools must be first loaded into the tool changer magazine. Tools will be called up and measured in sequential order.

Again, the setup person must make all axis movements manually, but once a tool has been brought to the position at which the tool length is displayed (this is where they would normally enter the value into the offset manually), the program will do so automatically.

- O9500 (Program to touch off tool lengths)
- N1 #100 = 1 (First tool station number to measure)
- N2 #3006 = 100 (TOUCH SPINDLE TO BLOCK)
- N3 #5003 = 0 (Set current Z position as program zero point)
- N4 G91 G01 Z1.5 F30.0 (Move away from block in Z)
- N5 G91 G28 Z0 M19 (Move to tool change position, orient spindle)
- N6 #101 = #100 (Counter for tool station number)
- N7 T#101 M06 (Place current tool in spindle)
- N8 #3006 = 101 (TOUCH TOOL TIP TO BLOCK)
- N9 #[2000 + #101] = #5003 (Set tool length compensation offset)

- N10 G91 G01 Z1.5 F30. (Move away from block in Z)
- N11 G91 G28 Z0 M19 (Move to tool change position, orient spindle)
- N12 #101 = #101 + 1
- N13 #3006 = 102 (PRESS CYCLE START TO CONT)
- N14 GOTO 7
- M30

Do not make setup people enter tool nose radius compensation offset values on turning centers
Entering data at the machine is always tedious. While control manufacturers do provide full access to *any* data that must be entered or modified by the setup person or operator, many do not make it convenient to do so. Additionally, entering data at the machine is time-consuming, error-prone – and is especially costly if the machine is not running production while entries are made.

Consider, for example, the task of entering CNC programs. While it is important that your people have the ability to *modify* programs during program verification, control manufacturers never intended this function to be used as the sole means for entering entire programs. Anyone entering a long program at the machine from scratch soon realizes that another method of loading programs must be found. Most companies, of course, create programs off line and use some kind of distributive numerical control (DNC) system to quickly load programs into a CNC machine.

Try to think of offset entries in the same way you think of program entry. You don't expect your setup people to enter lengthy programs during setup – but do you expect them to enter more than just a few offsets? Just like entering programs, manually entering offsets is tedious, time-consuming, and error prone.

If you know – or can determine – the value of an offset before a setup must be made, don't make the setup person manually enter it during setup. Look at doing so as being just as wasteful as making them enter CNC programs. Instead, find a way to *program* offset entries. The more offsets that must be entered, the more you'll benefit.

Most controls use a G10 command for offset entry. With one popular machining center control, for example, the command

- G90 G10 L1 P1 R5.0048

enters the value 5.0048 into offset number one. The command

- G90 G10 L2 P1 X-12.4363 Y-9.3762 Z-11.1827

enters the specified values into fixture offset number one.

In some applications, offset-entry commands can be included in the CNC program that machines the workpiece. To enter fixture offsets for repeated jobs having qualified setups, for example, the G10 command/s can be placed at the beginning of the program (though they'll be executed every time the program is run) – or you can place them at the end of the program (after the end of program command). Consider these commands:

- .

- .
- .
- N185 M30 (End of program)
- N999 G90 G10 L2 P1 X-12.4363 Y-9.3762 Z-11.1827
- G10 L2 P2 X-16.3734 Y-9.8373 Z-11.23736
- M30

During the setup, the setup person will search to line N999 and execute from there (like rerunning a tool. The G10 commands will be executed and the program will end. The N999 sequence will not be seen again (unless someone scans to line N999).

In similar fashion, you can include tool nose radius compensation offset entries (for turning centers) right in the machining program (at the beginning or end). These commands enter two sets of tool nose radius offsets, one for a turning tool and the other for a boring bar.

- O0001 (Program number)
- N005 G10 P2 R0.0316 T3
- N010 G10 P5 R0.0156 T2
- .
- .

In other applications, it is better to create a separate program to enter offsets. If you measure tool length compensation values for your machining centers off line, for example, the tool setter can create an offset entering program like this one.

- O8001 (Enter offsets)
- G90 G10 L1 P1 R6.2343
- G10 L1 P2 R8.3476
- G10 L1 P3 R6.9233
- G10 L1 P4 R5.3433
- G10 L1 P5 R7.3433
- G10 L1 P6 R8.7682
- G10 L1 P7 R9.2836
- G10 L1 P8 R7.6454
- G10 L1 P9 R8.3486
- G10 L1 P10 R5.1038
- M30

This program can be loaded during setup from the company's DNC system. When it is run once, all ten offsets for the job will be entered.

Is it worth it?

Your initial response to this suggestion may be that it will take someone just as long to type offset entries off line as it does the setup person to do so during setup. But remember, time spent performing setup-related tasks *off line* is not nearly as valuable as time spent while the CNC machine

is down during setup. Even if someone must work longer (or harder) to perform the task off line, it may still be beneficial to do so.

We've given three applications for programming offset entries. Can you think of others that can be used in your own company?

Simplify setup-related tasks: Program verification

Part of verifying a program is running the first acceptable workpiece. Here are some programming techniques that can help operators do so.

For machines that allow it, ensure cutting tools can be run

While sliding headstock lathes do not allow operators to (feasibly) rerun cutting tools, it is possible to rerun tools on machining centers and fixed headstock lathes. This give the operator the ability to size in the first workpiece, tool by tool. Trial machining (also called test cutting) can be done to ensure that each cutting tool machines all surfaces within tolerance bands before proceeding to the next tool.

In order for cutting tools to be rerun, certain commands necessary for the first tool must be programmed for every tool. Consider this example program:

- %
- O0002 (Program number)
- N005 G20 G18 G99
- N010 G50 S4000
-
- (ROUGH FACE AND TURN)
- N005 T0101 **M41**
- N010 G20 G96 S500 **M03**
- N015 G00 X3.2 Z0.005 **M08**
- N020 G99 G01 X-0.062 F0.012
- N025 G00 Z0.1
- N030 X2.33
- N035 G01 Z0
- N040 X2.58 Z-0.12
- N045 Z-1.495
- N050 X3.2
- N055 G00 X8.0 Z7.0
- N060 M01 (Optional stop)
- (DIAMETER SHOULD BE 2.580)
-
- (DRILL CENTER HOLE)
- N065 T0202 **M41**
- N070 G97 S1150 **M03**
- N075 G00 X0 Z0.1 **M08**
- N080 G01 Z-2.705 F0.008
- N085 G00 Z0.1
- N090 X8.0 Z7.0
- N095 M01
- (1" HOLE MUST BE 2.405 DEEP)
-
- (ROUGH BORE)
- N100 T0303 **M41**
- N103 G96 S400 **M03**
- N105 G00 X1.585 Z0.1 **M08**
- N110 G01 Z0 F0.007
- N115 X1.46 Z-0.0575
- N120 Z-1.87
- N125 X0.95
- N130 G00 Z0.1
- N135 X8.0 Z7.0
- N140 M01
- (HOLE DIAMETER MUST BE 1.460)
-
- (FINISH BORE)
- N145 T0404 **M42**
- N150 G96 S500 **M03**

- N155 G00 X1.625 Z0.1 **M08**
- N160 G01 Z0 F0.005
- N165 X1.5 Z-0.0625
- N170 Z-1.875
- N175 X0.95
- N180 G00 Z0.1
- N185 X8.0 Z7.0
- N190 M01
- (HOLE DIAMETER MUST BE 1.500)
-
- (FINISH FACE AND TURN)
- N195 T0505 **M42**
- N200 G96 S500 **M03**
- N205 G00 X2.7 Z0 **M08**
- N210 G01 X1.3 F0.005
- N215 G00 Z0.1
- N220 X2.25
- N225 G01 Z0
- N230 X2.5 Z-0.125
- N235 Z-1.5
- N240 X3.2
- N245 G00 X8.0 Z7.0
- N250 M01
- (OUTSIDE DIAMETER MUST BE 2.500)
-
- N250 M30 (End of program)
- %

If your initial experience with CNC programming is with sliding headstock machines (where it is not feasible to size in each tool in the manner just described), you may not consider doing so for other machines. But operators can ensure that the very first workpiece is acceptable if using trial machining techniques for machines that allow them.

Notice the bolded words in the previous program. You may think, for example, that the M41, M03, and M08 commands are not necessary in lines N065, N070, and N075 since the low spindle range is remains the same, and the spindle and coolant were not turned off since commands N005, N010, and N015. But these words are necessary if tool two is to be rerun.

If the operator runs the entire program before it is determined that tool two must be rerun, or if the setup person is trial machining, the spindle range may not be correct (later tools in the program select M42), and the spindle and coolant will have been turned off. This means the seemingly redundant words are required if tool two is to be rerun.

Provide obvious restart block for machines that allow operators to rerun cutting tools

We have made this point before, but it bares repeating. If operators are expected to rerun cutting tools, they must know where to place the cursor in the program before pressing the cycle start button. Our technique of skipping a line between tools should nicely suffice.

Anther way would be to use a special series of sequence number (N-words) to specify restart command. N9001 could always begin tool one, N9002 could begin tool two, N9003 for tool three, and so on.

Program trial machining operations

During setup, while running the first workpiece, the setup person must adjust all wear offsets in such a way that workpiece attributes are coming out at their target dimensions. This way, cutting tools will last for an extended period of time (possibly their entire lives) before a sizing adjustment is required. While this can be very challenging for sliding headstock machines (since the entire program is

commonly run before the workpiece can be measured), the process is simplified for other kinds of multi-tool metal-cutting machines. Each tool can be "sized in" before proceeding to the next tool.

The same process used to size in a cutting tool during setup must be repeated during production runs whenever dull tools are replaced.

It is common practice to use *trial machining* techniques to size in cutting tools that machine surfaces that have small tolerances. Trial machining involves adjusting the cutting tool's offset in such a way that excess material will be left on the machined surface. The cutting tool is then allowed to machine (again, allowing the excess material). The machine is then stopped and the setup person or operator measures the machined surface. They then adjust the offset by an amount that will bring the surface to its target value. Finally, the tool is rerun. This time the dimension will be very close to its target size. Any deviation will be related to the tool pressure difference when machining the trial machining stock versus the actual amount of stock left for machining.

While the concept of trial machining is not at all difficult to understand, trial machining does require manual intervention from the person running the machine. Anything you can do to simplify and streamline the process will, in turn, improve productivity.

Trial machining is only required, of course, when a new cutting tool machines a critical surface for the first time. So only finishing tools that machine small tolerances require trial machining. Once the machine is in production, the vast majority of machining cycles will not require trial machining – so if we're going to program the trial machining operation, we must provide a way for the setup person or operator to specify whether or not trial machining is currently required.

An easy way to do this is to use the block delete function. The operator will turn on the block delete switch when trial machining is not required (the machine will skip any trial machining commands). They will turn off the block delete switch when trial machining must be done.

Here is an example for sizing in critical diameters on a turned workpiece. Tool number one is the rough turning tool and tool number two is the finish turning tool. We're going to size in both tools together, ensuring that the appropriate amount of finishing stock is left on all diameters (we'll say 0.040 in)– and that tool pressure cannot affect the sizing adjustment. For this example, we can do the trial machining first, and in an area of raw material that will be eventually removed from the workpiece. This way there is no chance of removing too much material and scrapping the workpiece. We'll say the raw material stock diameter is 3.0 in, and the depth-of-cut for the roughing tool is 0.15 in.

- O0001
- (TRIAL MACHINING)
- /G10 P1 U0.01 (INCREASE ROUGHING OFFSET BY 0.01)
- /(ENSURES ROUGHER DOES NOT MACHINE TOO MUCH MATERIAL)
- /N005 T0101 M41
- /N010 G96 S400 M03
- /N015 G00 X2.7 Z0.1 M08 (TOOL WILL MACHINE TO 0.15 DEPTH)
- /N020 G01 Z-0.3 F0.015 (MACHINE ENOUGH TO TAKE MEASUREMENT)

- /N025 X3.1
- /N030 G00 X5.0 Z5.0 (MOVE TO TURRET INDEX POSITION FOR MEASURMENT)
- /N035 M00 (STOP FOR MEASUREMENT – DIAMETER SHOULD BE 2.7)
-
- /N040 T0101 (ENSURE THAT OFFSET CHANGE IS INSTATED)
- /N045 G00 X2.7 Z0.1 M08
- /N050 G01 Z-0.3 (ENSURE THAT DIAMETER IS TRULY 2.7)
- /N053 X3.1
- /N055 G00 X5.0 Z5.0
-
- /N060 T0202 M42 (INDEX TO FINISHING TOOL)
- /N065 G96 S500 M03
- /N070 G00 X2.62 Z0.1 M08 (TOOL WILL MACHINE 0.04 DEPTH)
- /N075 G01 Z-0.3 F0.008
- /N080 X3.1
- /N085 G00 X5.0 Z5.0
- /N090 M00 (STOP FOR MEASURMENT - TARGET IS 2.62)
-
- N095 T0101 M41 (ACTUAL MACHINING STARTS HERE)
- .
- .
- .

When trial machining is required, the setup person or operator will turn off the block delete switch. The G10 close to the beginning increases the rough turning tool's offset by 0.01 in, ensuring that this tool does not machine too deep on the first try. The roughing tool is then commanded to machine a small amount of material (at the tool's required depth-of-cut) and is retracted to the turret index position. The machine stops and the operator will measure the diameter. Since the diameter will be slightly over 2.700 in, they will adjust offset number one accordingly. Once the cycle is restarted, the roughing tool will machine again, ensuring that the diameter is 2.700.

The finishing tool will then machine 0.080 in more (in diameter) from the workpiece, retract to the turret index position, and stop. The operator will measure and if the diameter is not precisely 2.62, they will adjust accordingly. Machining of the actual part then begins. The roughing tool will leave precisely 0.04 in (on the side) and the finishing tool will machine precisely to programmed diameters.

Be sure to include a message after each program stop (M00) to tell the operator the size they are shooting for. Also, the entire series of trial machining commands could be placed in a subprogram, so only one command (the calling M98 command) need be in the main program. Also, this makes a pretty good application for a custom macro if you must perform trial machining operations on a series of similar workpieces.

A test part program for sliding headstock lathes

Since it is almost always necessary to run an entire workpiece on a sliding headstock lathe before the setup person can begin taking measurements and making sizing adjustments, it can be very difficult to figure out what must be done with offsets in order to get workpiece attributes to size. This

problem is further compounded by the sheer number of cutting tools that can be used on Swiss-type machines. It can be difficult to determine, for example, if one tool is machining too deep or another is machining too shallow.

This complexity often results in the need to run many workpieces to get one that passes inspection. This can be very frustrating, time consuming, and wasteful. Material costs may not be substantial (unless you run exotic materials), but the unpredictable amount of time it takes to get a part to pass inspection can be devastating, especially when you consider include the time it takes to *inspect* each workpiece before adjustments can be made.

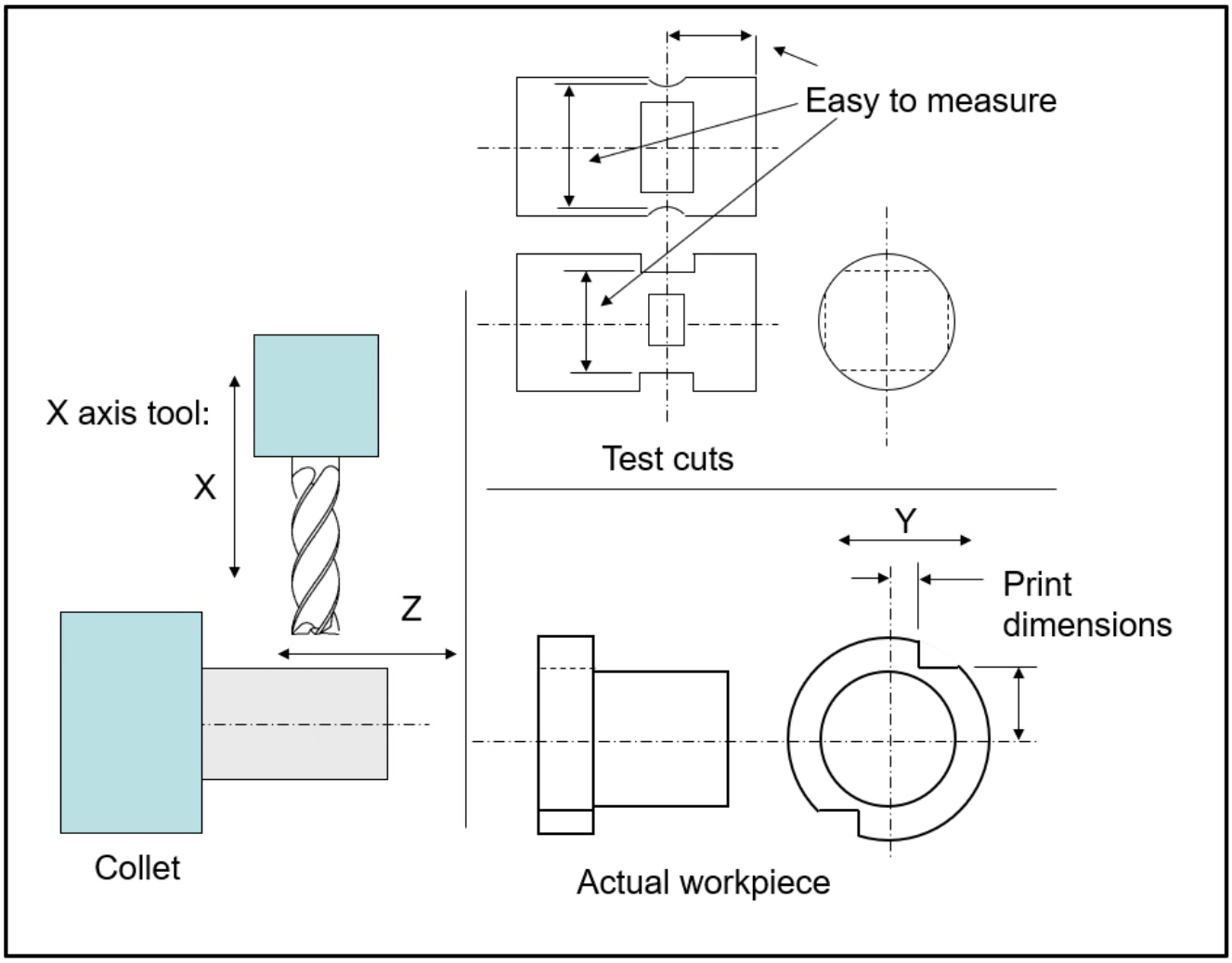
My suggestion is to develop a *universal test part program* that will be used during every setup to size in all of the cutting tools. The premise is that if a cutting tool is machining an attribute correctly on the test part, it will also machine actual workpiece attributes correctly. This, combined with the idea that each cutting tool will machine independent surfaces on the test part that are easy to measure, make's the test part program easy to use.

Frankly speaking, this idea been met with mixed reviews. Many programmers don't want to devote the time to develop a test part program. And many setup people feel that it is just as difficult to make a good test part as it is to make a good workpiece. While I still contend that the test part program will simplify and speed the process of running the first good workpiece, it is not of any value if no one wants it.

You still have to deal with the problems of frustration, lost time, wasted material, and unpredictable setup times. If your setup people are machining countless "practice workpieces" (I heard one manager say *buckets of them*) before they get one to pass inspection, I urge you to rethink your current methods.

Concentrate on the most difficult cutting tools. You can develop a series of test cuts for those cutting tools that give your setup people the most trouble. If you are doubtful about whether this method will work, start with *just one tool*. Once you are sure that this technique simplifies the needed adjustments as much as I say it will, you can always add more.

Here's an example. For many setup people, one troublesome type of tool is an end mill mounted parallel to the X axis. This kind of tool must often machine in the Y axis. With many machines, it can be difficult to perfectly center the end mill along the Y axis during setup. See the drawing for an example.



The test-part program will include commands to machine two sets milled surfaces – one flat and one rounded. The example program below machines them 0.050 smaller than the bar diameter. A comparator (or other measuring devices) can be used to easily determine if the Y axis is centered (and if not, how much it is off), if the tool is machining to the correct depth in X (bar diameter minus 0.05 inch), and if the tool is machining in the correct Z position (0.25 from part end to center of slot). With these values, the setup person can perfectly align the tool with offset adjustments and the Y axis program zero assignment before an actual part is run.

Here are the related programs. Notice that I am only showing motions – certain machine functions, like live tooling mode and main spindle mode selections are omitted.

The main program begins with some notes to point the setup person to the right place in the program. The cross-end mill portion of the program begins on line N005. This will allow you to use only one program for all test cutting operations. Under line N005, there are some variables for the setup person to specify. The program is then run from line N005 (in our example). An M30 ends each set of test cuts.

The program also uses your facing tool and cutoff tools. These tools are commonly kept in the machine on a permanent basis, so there won't be anything additional required to set them up. Each

segment of the program will face the part, perform the test cuts, and then cut off the part (about an inch long).

- O0001 (TEST CUTTING)
- (TEST CUTS INCLUDED IN THIS PROGRAM)
- (N005: CROSS END MILL)
- (N010: CROSS DRILL)
- (N015: THREAD MILL)
-
- N005 (CROSS END MILL)
- (SET VARIABLES)
- #100 =3 (TOOL STATION NUMBER)
- #101 =0.25 (END MILL DIA)
- #102 =0.5 (BAR DIAMETER)
- #103 =0.25 (Z POSITION FOR SLOTS)
- #104 =900 (RPM)
- #105 =2.75 (IPM)
- #106 =1 (FACING TOOL STATION)
- #107 =4 (CUTOFF TOOL STATNION)
-
- (DISTANCE ACROSS FLATS: BAR DIA MINUS 0.05)
- (DISTANCE BETWEEN ROUNDS: BAR DIA MINUS 0.05)
- (SLOTS CENTERED AT Z POSITION)
-
- (MACHINING)
- M98 P1000 (SUBPROGRAM FOR FACING)
- (MUST SELECT LIVE TOOLING MODE - M CODE?)
- T[#100*100+#100]
- C0 (START ON C ZERO SIDE)
- G98S#104 (SPEED FOR LIVE TOOL)
- G0Y[#102/2+.5+0.1]Z-#103 (LARGEST ENDMILL 0.5)
- X-0.25
- G1Y[#102/2-0.025+#101/2]F#105 (MILL ONE ROUND)
- G0Y[#102/2+.5+0.1]
- G0X[#102-0.05]
- G1Y-[#102/2+.5+0.1] F#105 (MILL ONE FLAT)
- G0X-0.25
- G1Y-[#102/2-0.025+#101/2] (MILL SECOND ROUND)
- G0Y-[#102/2+.5+0.1]
- X[#102+.2]

- C180
- X[#102-0.05]
- G1Y[#102/2+.5+0.1] (MILL SECOND FLAT)
- G0 X[#102+.2]
- Y0
- M98 P1001 (SUBPROGRAM FOR CUTOFF)
- M30
-
- N010 (CROSS DRILL)
- .
- .
- M30
-
- N015 (THREAD MILL
- .
- .
- M30
-
- O1000 (FACING SUBPROGRAM)
- T[#105+#105*100]
- (MUST SELECT MAIN SPINDLE MODE - M CODE?)
- G97S1000 M03
- G0X[#102+.1]Z0
- G1X-0.06F0.004
- G0X[#102+0.1]
- X1.0 Z0.1
- M99
-
- O1001 (CUTOFF SUBPROGRAM)
- T[#106+#106*100]
- (MUST SELECT MAIN SPINDLE MODE - M CODE?)
- G97S1000 M03
- G0X[#102+.2]Z-1.0
- G1X-0.125F0.004
- G0X[#102+0.2]
- X1.0 Z0.1
- M99

Deal with raw material Z surface variations in your machining center programs

You know that to truly qualify a program (keep it from ever having to be changed in the future), you must eliminate those things that change from one time the job is run to the next. But certain

variables may be beyond your control. If, for example, you run a cast iron often, it is likely that you experience raw material size variations on a regular basis.

Consider, for example, face milling a surface. If the surface is cast, and if the amount of material to be machined varies from one time the job is run to the next, the number of passes the face mill makes will have to be changed. Unfortunately, this is one of those times when may be constantly chasing your tail. The next time the job is run, the amount of stock on the surface may vary again, requiring yet another program change. Indeed, every time the job is run, you may have to make a program change. You may even experience this kind of variations within one lot of castings, meaning the program may have to be changed several time even during a single production run!

If it is not feasible to keep this surface from varying (the casting vendor may be beyond your control), the next best thing is to make it as easy as possible to change the program to adapt to the varying surface. With any version of parametric programming, you can make it so simple that any setup person will be able to quickly and easily deal with the variation.

At the beginning of your program, include two variables:

- O0001 (Main program)
- #120 = 0.25 (STOCK ON CAST SURFACE)
- #121 = 0.1 (Z DEPTH PER MILLING PASS)
- N005 T01 M06 (Normal beginning of program)
- .
- .

We've used variable #120 to specify how much stock is on the surface and #121 to specify the Z depth the maximum depth the milling cutter will take per pass.

Once the milling cutter is placed in the spindle and brought to its approach position in XY, give these commands to specify how milling will be done:

- N040 G43 Z1.015 (Rapid in Z to within one inch of final surface position)
- (0.015 value allows finishing stock for another cutter)
- N045 G65 P9001 S#120 D#121 C2001.0 (Mill surface)

Our technique requires that you rapid the tool in Z to within precisely one inch of the final Z surface you wish this rough milling cutter to mill. Note that we're going to be leaving 0.015 finishing stock with this rough milling cutter.

In line N045, we are calling custom macro O9001. S (set to #120 above) specifies the amount of stock in Z to be removed. D specifies the depth per pass (#121 above). And C specifies the program number for a subprogram that includes the XY motions for each pass. Note that our XY program (O2001 in this example) can include any conceivable set of motions, so there is no limit to what kind of movements can be done. But you do have to break them off into another program. Here is an example for the XY motion program:

- O2001 (Program containing XY motions)
- N001 G01 X5.75 F4.0
- N002 Y3.75
- N003 X0.25
- N004 Y-0.6
- N005 M99 (End of program)

Now, here is the custom macro that makes the multiple passes:

- O9001 (Program number)
- #110 = #5001 (Attain current abs position in X)
- #111 = #5002 (Attain current abs position in Y)
- #112 = #5003 (Attain current abs position in Z)
- #102 = FUP [#19/#7] (Calculate number of passes)
- #103 = #19/#102 (Recalculate depth per pass)
- #104 = 1 (Pass counter)
- #105 = #112 – 1.0 + #19 - #103 (Calculate Z position for current pass)
- N9001 IF [#104 GT #102] GOTO 9005 (If finished, exit)
- G00 Z#105 (Rapid to current Z surface to mill)
- G01 (Instate cutting mode)
- G65 P#3 (Make XY motions for this pass)
- G00 Z#112 (Retract to initial Z position)
- X#110 Y#111 (Move to initial XY position)
- #104 = #104 + 1 (Step counter)
- #105 = #105 - #103 (Step Z surface to mill)
- GOTO 9001 (Go back to test)
- N9005 M99 (End of custom macro)

Admittedly, this technique may at first appear to be a little complicated to program. But remember, the goal is to make it as quick and easy as possible for the *setup person or operator* to adapt to casting size variations. Frankly speaking, it couldn't get much easier for them. They will simply change one or two values at the beginning of their program!

Also, program O9001 is universal. It will work for any Z surface in any program, meaning the leg work of proving it out will only be necessary the first time you use it. From then on, it can remain in your control on a permanent basis (remember that you can even protect programs in the O9000 series from accidental editing

Deal with raw material X/Y surface variations in your machining center programs

The previous topic provided a way to help your setup people and operators deal with Z surface raw material variations. You may have been wondering if the same kind of technique could be applied to XY surface variations. It can. Here is an example cutting program that uses the technique.

- O0001 (Main program)

- #120 = 0.2 (STOCK IN CORED HOLE ON THE SIDE)
- #121 = 0.1 (DEPTH OF CUT IN XY)
- N005 T01 M06 (1" end mill)
- N010 G54 G90 S500 M03 T02
- N015 G00 X3.75 Y1.5 (Approach XY position, 0.5 from final surface to mill)
- N020 G43 H01 Z-0.85 (Rapid to work surface in Z)
- (Tool's periphery is 0.5 from final XY contour and at work surface in Z)
- N025 G65 P9002 S#120 D#121 C2002.0 T31.0
- N030 G00 Z0.1 (Retract to above workpiece in Z)
- N035 G91 G28 Z0 M19 (Retract to tool change position)
- N040 M01 (Optional stop)
- N045 T02 M06 (Tool change
- .
- .

There are three notable differences from the multiple Z surface custom macro application. First, the variable included at the beginning of the cutting program will now be specifying the amount of stock in XY (not Z).

Second, the prior positioning (approach) position in the main program must be to within 0.5 inch of the final pass in XY and to the work surface in Z. We're assuming there will never be more than 0.5 inch raw material to be removed.

Third, the call statement to the multiple pass custom macro includes the offset register number in which the cutter radius compensation value for this tool is stored (with T). The custom macro is going to use this value to come up with a temporary value (in offset number 99) that keeps the tool the appropriate current distance away from the final cut surface. Note that we're using cutter radius compensation to keep the tool away from the surface being machined, so you must confirm that the rules of cutter radius compensation are not broken with the largest offset value (this may present problems with small inside radii).

The movements needed for the final pass will be stored in a subprogram (O2002 in our example). Let's look at this subprogram for our example.

- O2002 (XY movements)
- N001 G01 G41 D99 X4.75 F4.0 (Instate cutter radius compensation to surface, the three o'clock position of the circle to mill)
- N002 G02 I-1.25 (Mill complete circle)
- N003 G00 G40 X3.5 Y2.5 (Cancel cutter radius compensation during move back to center)
- N004 M99 (End of program)

In the first command of this program, you'll be instating cutter radius compensation with offset 99. The multiple XY pass custom macro will calculate the appropriate value of this offset for each pass. Note that we're milling an inside contour of a cored hole and have no need to make moves to clear

obstructions after each pass. If milling an outside contour, be sure you leave the tool clear of obstructions (without moving in Z) prior to ending this program.

The multiple XY pass custom macro:

- O9002 (Custom macro to make multiple XY passes)
- #110 = #5001 (Retrieve current X abs position)
- #111 = #5002 (Retrieve current Y abs position)
- #102 = FUP [#19/#102] (Calculate number of passes)
- #103 = #19/#102 (Recalculate depth of each pass)
- #104 = 1 (Pass counter)
- #2099 = #[2000 +#20] + #19 - #103 (Value of cutter radius compensation offset for current pass is stored in offset number 99)
- N9001 IF [#104 GT #102] GOTO 99 (If finished, exit)
- G00 X#110 Y#111 (Move to starting position in XY)
- M98 P#3 (Make current XY pass)
- #104 = #104 + 1 (Step pass counter)
- #2099 = #2099 - #103 (Step current cutter radius compensation offset value)
- GOTO 9001 (Go back to test)
- N99 M99 (End of program)

This custom macro is determining how many passes to make based upon what the setup person specifies at the beginning of the main program. It then determines the initial value of offset number 99. Finally, it makes each pass, reducing the value of offset number 99 foe each successive pass by the appropriate amount.

One more point that applies to both the multiple Z pass and multiple XY pass custom macros. There may be times when you have more than one surface that is varying on a given workpiece (in either XY or Z). Keep in mind that you can handle this problem by adding input variables at the beginning of the main program.

Notice that in line N025 of program number O0001 (the main program), we specify the amount of stock on the surface (S) and the maximum depth of each pass (D) with variables. If you have more than one surface that is varying, and if they're varying by different amounts, simply reference the appropriate variable numbers (like #122, #123, and so on) whenever you command the execution of your multiple pass custom macros.

Simplify production run tasks

Just as there are things you can do with programs that help people during setup, so are there ways your programs can help people during production runs. Here are some examples.

Automate sizing adjustments for long running jobs

The primary function of any tool life management system is to extend the period of unattended operation for a machine tool. One facet of how extended unattended operation is achieved is

through the use of multiple identical cutting tools which are stored in the machine's automatic tool changing system or turret. The tool life management system will, by one means or another, determine when a cutting tool is dull. A period specified in time or number of parts allows it to make this determination. Once a tool is dull, it will automatically start using a fresh one. The system will also alert the operator so they will know a dull tool must be replaced.

An often-overlooked, yet equally important feature of any tool life management system is the ability to automatically make sizing adjustments to deal with tool wear during each cutting tool's life. As some tools dull, of course, the workpiece attribute/s they machine will change. This is especially true with single point cutting tools, like turning tools and boring bars used on turning centers, and is most critical for finishing tools, since they directly control workpiece sizes. A tool life management system will provide some way to make sizing adjustments based upon the user's understanding of how tools will wear during their lives.

Unless a CNC turning center is equipped with an automatic tool changing system (the vast majority are not), they do not typically have enough tool stations to allow the multiple cutting tools needed to make a tool life management system feasible. The turrets for most turning centers can hold up to only twelve tools; even less if some turret stations must be left empty to deal with interference problems.

While multi-tool based applications for tool life management systems may not be practical for turning centers, the sizing adjustment aspect does remain feasible. Due to the nature of how single point finishing tools wear, almost every turning center user has probably wished at one time or another that they could automatically make offset sizing adjustments, especially for longer production runs.

Consider, for example, a CNC lathe (fixed or sliding headstock) that is equipped with a multi-bar bar feeder. If not for cutting tool dulling issues, these machines could run unattended for hours – if not days. While you may not be able to automate the replacement of dull tools, you *can* automate the task of making sizing adjustments, if you know what happens to a cutting tool during its life.

Most CNC users will not purchase an expensive tool life management system just to use the sizing adjustment feature. Fortunately, you do not have to. A simple custom macro can handle your sizing adjustment, again, if you know when sizing is needed and how much adjustment is required.

Here is the skeleton of a main program that uses the custom macro:

- O0001(Machining program)
- .
- .
- N075 T0202 (Finish turning tool)
- .
- .
- .
- T0505 (Finish boring bar)
- .

- .
- .
- N335 G65 P1000 T2.0 C500.0 A-0.0001 F1200.0 S1.0 (Finish turning tool sizing)
- N340 G65 P1000 T5.0 C60.0 A0.0002 F700.0 S2.0 (Finish boring bar sizing)
- N345 M30 (End of program)

At the end of the program, the two G65 commands call the sizing custom macro. T is the tool station number, C is the number of workpieces machined before an adjustment is required. A is the amount and polarity of adjustment, F is the total number of workpiece the cutting tool can machine before it's dull, and S sets the counter number (two permanent common variables will be used for counting – in this example, #501 and #502 are used).

In line N335, we're specifying that the finish turning tool is in station two, and will last for fifty parts before a 0.0001 inch sizing adjustment must be made. This will continue until the machine has machined twelve hundred parts – at which time the cutting tool is dull and must be replaced. Something similar is happening for the finish boring bar in line N340.

Local variable representations in the custom macro:

- T: #20, C: #3, A: #1, F: #9 S#19

Here is the custom macro. Note that it even stops the machine and resets the wear offset (to zero) when a tool is dull.

- O1000 (Offset adjusting program)
- #[500+#19] = #[500+#19] + 1 (Dull tool counter)
- #[510+#19] = #[510+#19] + 1 (Adjustment counter)
- IF[#[500+#19] LT #9] GOTO 5 (Is the tool dull? No: go to N5)
- #[500+#19]=0 (Reset dull tool counter)
- #[510+#19] = 0 (Reset adjustment counter)
- #100 = #20 (Operator can check #100 to see which tool is dull)
- G10 P[#20] X0 (Reset wear offset to zero)
- #3000 = 100 (CHANGE INSERT)
- N5IF[#[510+#19] LT #3]GOTO 10 (Is an adjustment needed? No: go to N10)
- #[510+#19] = 0 (Reset adjustment counter)
- G10 P#20 U#1 (Make sizing adjustment in wear offset)
- N10 M99

Keep track of how long tools last before the must be replaced

For repeated jobs, especially long running ones, be sure operators know when cutting tools will get dull. This should be pretty easy once you have some experience running the job. Track how many parts can be machined before tools need to be replaced and include a message at the beginning of each tool (or in the production run documentation) that specifies this important information.

- %
- O0002 (Program number)
- N005 G20 G18 G99
- N010 G50 S4000
-
- (ROUGH FACE AND TURN)
- (REPLACE INSERT AFTER 350 PARTS)
- N005 T0101 M41
- N010 G20 G96 S500 M03
- N015 G00 X3.2 Z0.005 M08
- N020 G99 G01 X-0.062 F0.012
- N025 G00 Z0.1
- N030 X2.33
- N035 G01 Z0
- N040 X2.58 Z-0.12
- N045 Z-1.495
- N050 X3.2
- N055 G00 X8.0 Z7.0
- N060 M01 (Optional stop)
- (DIAMETER SHOULD BE 2.580)
-
- (DRILL CENTER HOLE)
- (REPLACE INSERT AFTER 500 PARTS)
- N065 T0202 M41
- N070 G97 S1150 M03
- N075 G00 X0 Z0.1 M08
- N080 G01 Z-2.705 F0.008
- N085 G00 Z0.1
- N090 X8.0 Z7.0
- N095 M01
- (1" HOLE MUST BE 2.405 DEEP)
-
- (ROUGH BORE)
- (REPLACE INSERT AFTER 350 PARTS)
- N100 T0303 M41
- N103 G96 S400 M03
- N105 G00 X1.585 Z0.1 M08
- N110 G01 Z0 F0.007
- N115 X1.46 Z-0.0575
- N120 Z-1.87
- N125 X0.95
- N130 G00 Z0.1
- N135 X8.0 Z7.0
- N140 M01
- (HOLE DIAMETER MUST BE 1.460)
-
- (FINISH BORE)
- (REPLACE INSERT AFTER 500 PARTS)
- N145 T0404 M42
- N150 G96 S500 M03
- N155 G00 X1.625 Z0.1 M08
- N160 G01 Z0 F0.005
- N165 X1.5 Z-0.0625
- N170 Z-1.875
- N175 X0.95
- N180 G00 Z0.1
- N185 X8.0 Z7.0
- N190 M01
- (HOLE DIAMETER MUST BE 1.500)
-
- (FINISH FACE AND TURN)
- (REPLACE INSERT AFTER 500 PARTS)
- N195 T0505 M42
- N200 G96 S500 M03
- N205 G00 X2.7 Z0 M08
- N210 G01 X1.3 F0.005
- N215 G00 Z0.1
- N220 X2.25
- N225 G01 Z0
- N230 X2.5 Z-0.125
- N235 Z-1.5
- N240 X3.2
- N245 G00 X8.0 Z7.0
- N250 M01
- (OUTSIDE DIAMETER MUST BE 2.500)
-
- N250 M30 (End of program)
- %

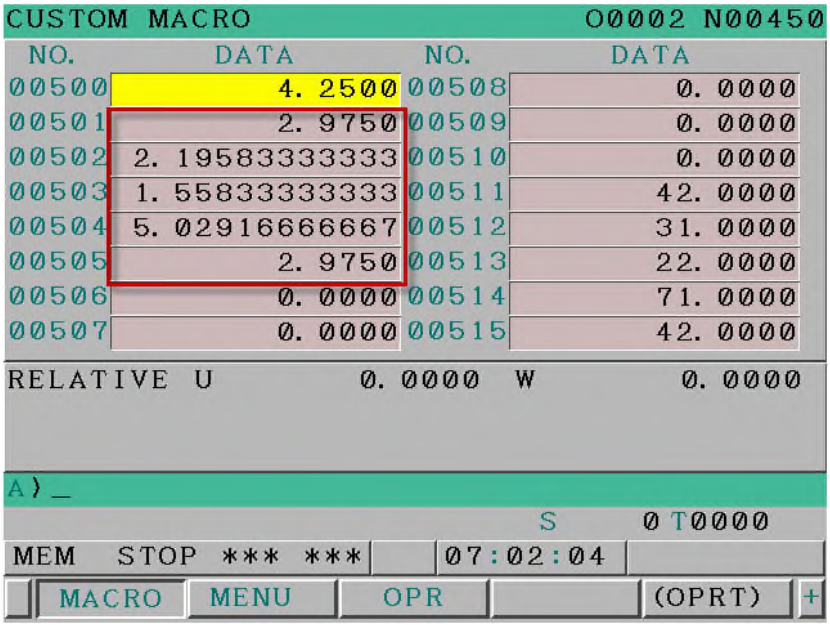
A program that tells operators when tools will get dull

Most tool life management systems are good at telling you when a cutting tool is dull, but some are not so good at predicting *when* each tool will become dull in the future. This Custom-Macro-based tool life manager overcomes this limitation.

It will help with automated machines, like turning centers equipped with bar feeders, that allow long periods of unattended operation. Maybe the operator will be doing other things while a job is running, but wants to know when to return in order to perform tool maintenance. Or maybe the job is going into a long period when no one will be available to perform tool maintenance (overnight) and the operator wants to see which tools must be replaced before they leave.

This tool life management system contains four programs. Program number O0100 (which can be renumbered and saved with each job) is the data entry program. In it you specify the cycle time including part loading and the number of cycles for which each tool will last before it gets dull. You also specify the number of tools being monitored (up to ten). Program O0001 is the main program, the one used to machine workpieces. Program O9500 will reset the tool life data for tools after replacement. Program O9501 is the tool life monitoring program.

The operator will monitor permanent common variables #501 through #510 to see how much longer (in hours) each tool will last before getting dull.



If an operator is about to leave the machine to do something else, these permanent common variables will tell her when to return. Or if no one will be available for a long time (possibly overnight), she will be able to tell which tools must be replaced before leaving. If a tool is replaced before it is dull (again, to achieve the longer period of unattended operation), she simply sets the related permanent common variable (#501-#510) to zero and the tool's life will be updated when the next cycle runs.

To use these Custom Macros, begin the job with a new set of tooling/inserts. Modify program O0100 to specify cycle time, the number of workpieces for which each tool will last, and the number of tools in the job. If one of the tool stations is not used in the job (maybe tool number two is an empty

station, for example), set its related variable to a number greater than the number of workpieces in the production run. When finished, run this program once. (You now can monitor permanent common variables #501-#510 to see the current time remaining for each tool.)

Next, modify your machining (main) program to call Custom Macro O9500 at the beginning and O9501 at the end. Finally, start running production. When a tool is dull, an alarm will sound. Look at variables #501-#510 to determine which tool/s is/are dull (one or more of variables #501-#510 will have a value of zero) and do the related tool maintenance. Reset the program to continue.

Any time you want to see how much longer each tool will last, look at variables #501-#510. If you do decide to replace tools before they are dull (again, to achieve a longer period of unattended operation) remember to set the related #501-#510 variable/s to zero.

Here are the custom macros:

- **O0100** (DATA ENTRY AND INITIALIZING)
- #500 = **4.25** (CYCLE TIME IN MINUTES - INCLUDE LOADING)
- #521 = **50.0** (TOOL 1 NUMBER OF CYCLES)
- #522 = **70.0** (TOOL 2 NUMBER OF CYCLES)
- #523 = **90.0** (TOOL 3 NUMBER OF CYCLES)
- #524 = **120.0** (TOOL 4 NUMBER OF CYCLES)
- #525 = **100.0** (TOOL 5 NUMBER OF CYCLES)
- #531 = 5 (NUMBER OF TOOLS BEING MONITORED, MAX 10)
- (DO NOT MODIFY BELOW)
- #1=1
- N1 IF[#1 GT #531] GOTO 99
- #[510 +#1] = #[520 +#1]
- #[500 +#1] = #[510 +#1] * #500/60
- #1=#1+1
- GOTO 1
- N99 M30

- **O0001** (MACHINING/MAIN PROGRAM)
- M98 P9500 (RESET TIMES IF NECESSARY)
- ()
- (ALL MACHINING HERE)
- ()
- M98 P9501 (CHECK TOOLS)
- N450 M30

- **O9500** (RESET TIME IF NECESSARY)
- #1=1
- N25 IF[#1 GT #531] GOTO 99
- IF [#[500 +#1] LE 0] THEN #[510+#1] = #[520+#1]
- IF [#[500 +#1] LE 0] THEN #[500+#1] = #500 * #[520+#1]
- #1=#1+1
- GOTO 25
- N99 M99

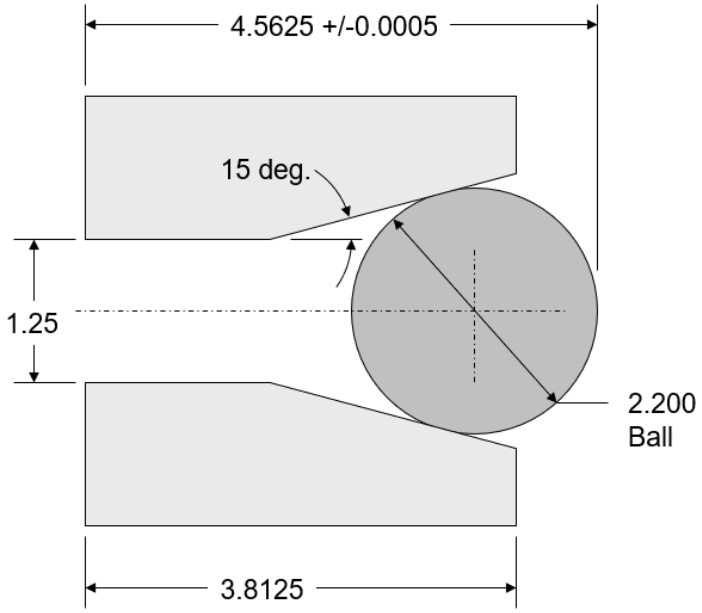
- **O9501** (CYCLE COUNTER & TIME CHANGER)
- #1=1
- N5 IF[#1 GT #531] GOTO 10
- #[510+#1] = #[510+#1] - 1
- #1=#1+1
- GOTO 5
- N10
- #1=1
- N12 IF[#1 GT #531]GOTO 13
- #[500+#1] = #[510+#1] * #500/60
- #1=#1+1
- GOTO 12
- N13
- #2=0
- #1=1
- N15IF[#1 GT #531] GOTO 20
- IF[#[500+#1] LE 0] THEN #2=1
- #1=#1+1
- GOTO 15
- N20
- IF [#2 EQ 0] GOTO 99
- #2=0
- #3000 = 100 (REPLACE DULL TOOL/S)
- N99 M99

Eliminate calculations needed to make sizing adjustments

Most sizing adjustments a turning center operator makes are pretty easy, requiring no more than simple addition and subtraction. If a diameter is coming out 0.002 in too large, the X axis offset for the cutting tool machining the diameter is reduced by 0.002 in. Similar techniques are used for length (Z axis) dimensions.

But consider the workpiece shown in the drawing below. The operator must place a gauge ball in the tapered bore and measure workpiece length over the ball. This time simple arithmetic will not suffice

for the adjustment should the dimension need adjustment. Indeed, the taper's starting X position and ending Z position must be altered, requiring right angle trigonometry.

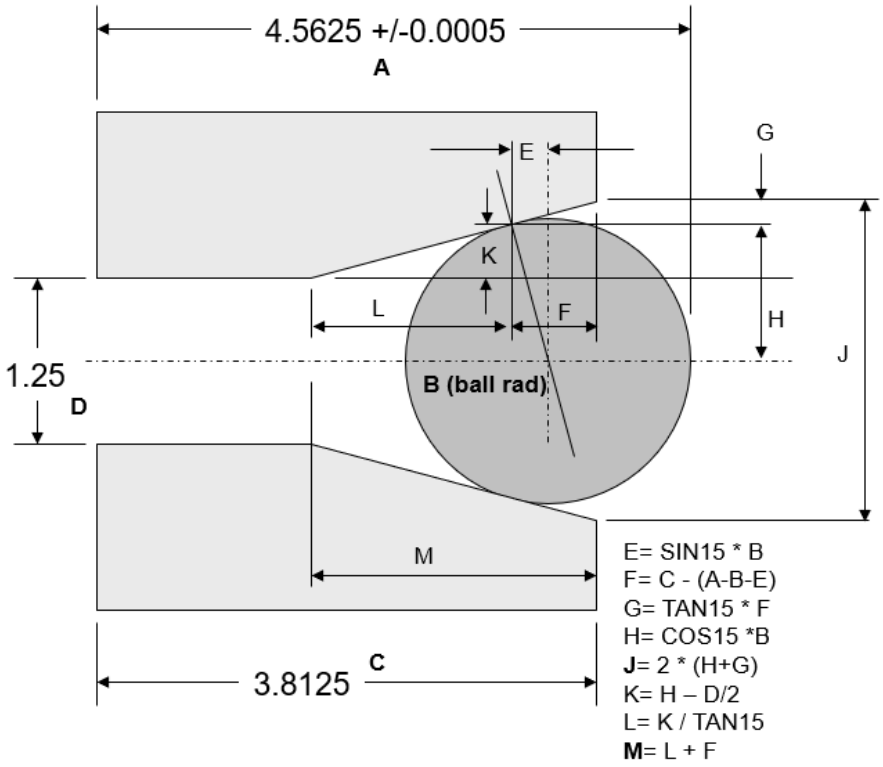


The problem is compounded because this dimension has a very tight tolerance. Several sizing adjustments may be required during a cutting tool's life as the tool wears.

You can simplify the task of making complex sizing adjustments using custom macro. We are showing a specific example, but consider any time when your operators struggle when making sizing adjustments. It is likely that similar techniques can be used.

In our solution, the operator will use a permanent common variable (#501) to specify the deviation from the required dimension (4.5625) to the measured dimension. If the dimension comes out 0.004 oversize, they will enter 0.004 in permanent common variable #501. If it comes out undersize by 0.003, they will enter -0.003. Note that it may be more convenient to use an offset register in which to enter this deviation. If it makes more sense to do so, by all means, do it. Just determine the appropriate system variable number to access the desired offset register.

Here is a drawing that shows the math calculations required:



Below is the segment of the custom macro program that handles this issue. Only the finish boring bar commands are shown. You can surely come up with a more elegant way to handle the math, but we've made it match the drawing for clarity.

- O0001
- #502=4.5625-#501 (Distance over ball- A)
- N3 #1= SIN[15]*1.1 (E)
- #2= 3.8125 - [#502 - 1.1 -#1] (F)
- #3= TAN[15] *#2 (G)
- #4= COS[15] * 1.1 (H)
- #5= 2 * [#4 + #3] (J)
- #6= #4 - 1.25/2 (K)
- #7= #6 / TAN[15] (L)
- #8= #7 + #2 (M)
-
- T0202 M42
- G96 S500 M03
- G00 X[#5 +2*TAN[15]*.1] Z0.1 M08 (0.1 in approach distance)
- G01 X1.25 Z-#8 F0.008
- Z-4.9
- X1.15
- G00 Z0.1
- X7.0 Z8.0
- M30

Again, if the distance over the ball is coming out precisely on size, permanent common variable #501 will be zero. (You must confirm that #501 is set to zero before the production run is started.) If the 4.5625 dimension comes out to 4.5635, the operator will enter 0.001 in permanent common variable #501. If it comes out to 4.5613, the operator will enter -0012 in #501. In either case, the custom macro will calculate the appropriate start point in X and end point in Z for the taper move.

Make it easy for operators to determine which measuring device to use and which offset to adjust

Companies vary when it comes to how they utilize CNC people. In some companies, like contract shops, one person is commonly responsible for the entire CNC job, including programming the job, setting it up, and running production. This person is, of course, very familiar with the job, and will know which cutting tools machine each workpiece surface.

In other companies, like many product producing companies, one person programs the job, another sets it up, and yet another runs production. The person running production is commonly called the CNC operator. For large lots, several CNC operators may be involved (first shift, second shift, third shift – for example).

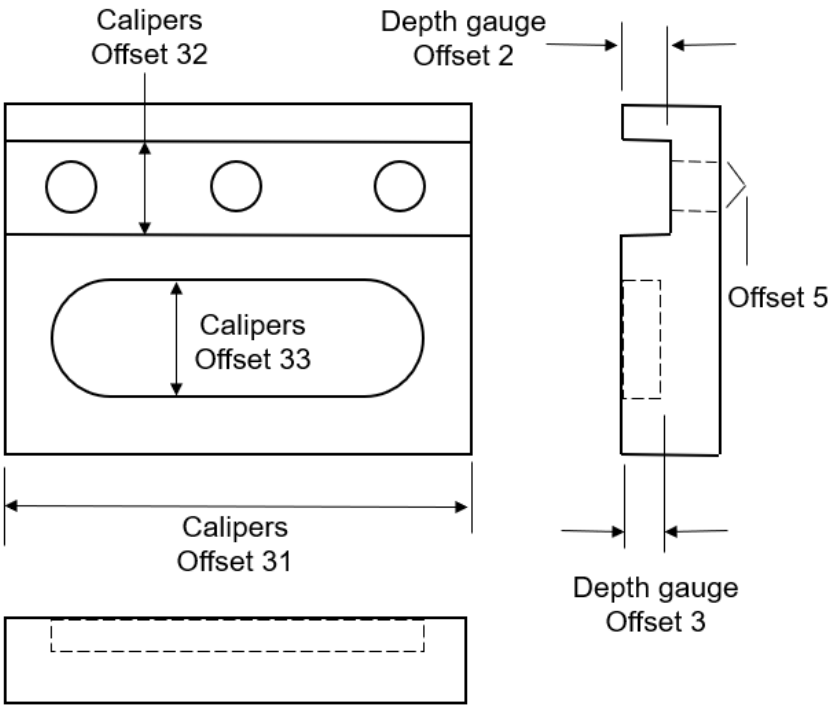
A CNC operator will not be nearly as familiar with a job as the programmer. Yet many programmers provide minimal (if any) production run documentation to help CNC operators know what they're supposed to do during a production run. They depend upon word-of-mouth to answer any questions that CNC operators may have. One especially troublesome area has to do with sizing adjustments that are required during long productions runs.

When a workpiece surface is growing (or shrinking) due to tool wear, and as it nears a tolerance limit, the CNC operator must make a sizing adjustment. To do so, they must know which offset controls the workpiece surface in question. Since an offset is specified by a number, and since the offset number is usually made to correspond in some way to the tool station number, the CNC operator must know which cutting tool machined the surface (and its tool station number) in order to determine which offset must be adjusted.

While this may sound like a simple task to an experienced programmer or setup person that is intimately familiar with the job, it can be quite challenging for CNC operators. The more tools used in the job, the more challenging it will be. Consider a twenty-tool machining center program. If the depth of a pocket must be adjusted, the CNC operator may take several minutes to determine which milling cutter in the automatic tool changer magazine machines the pocket.

Most setup documentation will not help. While the setup sheet will provide a list of cutting tools and station numbers, nothing ties cutting tools to the workpiece surfaces they machine. And, of course, if the operator makes a mistake and changes the wrong offset, the results can be disastrous.

You can dramatically simplify the task of determining which offset must be used to make a sizing adjustment in your *production run documentation*. Consider this drawing:



We're showing a machining center application, the same techniques can be used for turning center applications. Notice that every machined surface is documented. This documentation names the gauging tool must be used to measure each workpiece surface, as well as the offset number that must be adjusted when sizing is required.

Safety

Of course you want your CNC programs to be safe to run. You do not want to be doing anything that would put the operator in danger. Neither do you want to do anything that could cause damage to the machine. And finally, you want your programs to make good parts. So, the three priorities for safety are operator safety, machine safety, and workpiece safety. Here are some suggestions for avoiding dangerous situations.

Machine safety

Here are a few issues that I've seen over the years that can cause or worsen machine problems.

Not knowing your M-codes could lead to machine longevity issues

Machine tool builders vary dramatically when it comes to what they allow you to do with M codes. Some are quite limited in this regard while others allow you to control almost everything by M codes. Too many times the programmer ignores all but the most basic M codes when working with their particular CNC equipment. They don't even consider the possibility that an M code may be available to help with a particular application. This is especially true when a new programmer takes over the programming of an existing piece of CNC equipment.

One of the first things you should do whenever you begin working with ANY machine that is new to you is to study the list of M codes available for the machine you will be working with. You will

normally find the list of M codes supplied in the machine tool builder's manual (not the control manufacturer's manual). You may be surprised at what you find. For any M codes that do not make sense to y2u, contact your machine tool builder to learn their usage.

Neglecting your M codes list can cause problems ranging from inconvenient to downright damaging to the machine tool. For example, on a turning center you may find you have control of your chucks clamping direction (internal or (external) by M code. This would make it possible to easily program for first and second operations on a workpiece within the same program (turning the part around after the first end is finished) in conjunction with outside diameter and inside diameter chucking. Maybe the first operation chucks on the outside of the workpiece and the second chucks in the inside. Without knowing these M codes exist, you wouldn't even consider this a feasible possibility!

In more extreme cases, if you don't know your M codes, you could cause damage to the machine. For example, say a company has a horizontal machining center with a rotary (B) axis incorporated in the table – or a turning center that has a rotary (C) axis in the spindle. For these machines, say the programmer is supposed to specify an M code to clamp the table before performing any powerful machining operations on a particular side of the table (though not all rotary axes require this). Not doing so would place undue strain on the rotary axis drive motor and drive mechanism, and this system will wear out much faster than it should have.

Applications for M codes

The primary application for M codes is to allow programmable on/off switches for mechanical devices. This allows the programmer to activate mechanical functions like spindle, coolant, chuck jaws, hydraulic clamping, tailstock, and indexers. Again, some machine tool builders give the programmer control of almost everything (even chip conveyors, work lights, and protective doors) while others are quite weak in this regard.

A second application for M codes is to allow programmable on/off software related switches. Usually these functions are related to some feature of programming. For example, some machine tool builders allow the programmer to control whether optional block skip (or block delete) is on or off through M codes. M12 may be used to turn on this feature and M13 to turn it off. Other applications for M codes in this category include enabling and disabling feedrate override, controlling thread chamfering for turning centers, controlling how cutter radius compensation is handled for rounding corners, and controlling whether mirror image is turned on or off.

Some rather obscure M codes

As you look through your M codes list, be prepared for some rather obscure functions. For any you do not understand, be sure to contact your machine tool builder. You may find for example, you have control of whether an M code is activated at the beginning of a motion command or whether the M code will not be activated until the end of the motion command. You may find you have control of whether the machine will turn itself off at the completion of the program (for unattended use). You may have control of how many "look ahead" commands the control will use with cutter radius compensation.

These are but a few of the rather unusual uses for M codes. It may take quite a bit of study to understand the implications of when these code can help. Be sure you understand these implications.

Confirm all initialized states

Programmers often assume quite a bit of their CNC machines when they execute a program. They may, for example, assume that certain initialized states are still set the same as they were when the machine's power was turned on. If this is not the case, the program will not run as expected. For example, maybe they assume that the measurement system of choice (Imperial or Metric) is still selected when the program is run. If it is not, the axes will not move as expected.

To avoid certain assumptions, you may be in the habit of including a series of initializing commands at the beginning your programs to ensure that initialized states are still active. You could, for instance, include a G20 (Imperial) or G21 (Metric) at the beginning of all programs to ensure that the appropriate measurement system has been selected.

While this is a very good habit, one that I would urge all programmers to consider, including initializing commands in all of your programs can be cumbersome. And if at some future date you discover another one that must be added, you'll probably have a lot of programs to change.

My suggestion is to create a user defined G code (like G100) with which to specify your program initializing commands. In this way you can store all of your initializing words and commands in one place. If you must eventually add more, you can easily do so without having a lot of programs to change. This assumes your machine is equipped with Custom Macro. If not, the same result can be accomplished with a simple sub-program call.

Include more than just initializing G codes

Just as an inappropriately selected G code will cause your programs to behave poorly, so can an inappropriately specified parameters. Along with initialized G codes, you can ensure that program-relate *parameters* are set as they must be when your programs are executed. If a given parameter varies among your programs - or machines (maybe the parameter that controls the retract amount for a chip-breaking peck drilling cycle) - you can even include variables if you create a user defined G code, and specify the currently required value.

Parameter setting commands would be overly cumbersome to include in all of your CNC programs, but they are quite easy to include in your special initializing user defined G code (G100) program.

Including parameter settings in your initializing program provides a somewhat transparent benefit (as opposed to including them in your machining program). Since parameter numbers vary among control models, you may require a different initializing program in each machine. Each will appropriately set your chosen parameters. This cannot be accomplished if you elect to include all initializing commands in the machining program and you expect the program to run on different machines.

To create a user defined G code, with a FANUC 0iD control, you can set parameter number 6050 to a value of 100. After that, the control will execute program number O9010 whenever it reads a G100 word. Initializing commands will be included in program O9010

An example

Maybe you have had issues with program format when using multiple repetitive cycles on a turning center. Newer FANUC controls allow both one-line as well as two-line multiple repetitive cycles. A parameter controls which program format must be used. Wouldn't it be helpful to ensure that this parameter is appropriately set when programs are run? If it is not, the control will generate an alarm when your program is run. It will take time to diagnose and correct the problem. (By the way, this is but one example of many program-related parameters that can be included in your initialing program.)

Parameters vary among FANUC control models, so you must reference your parameter manual to find program-related parameter/s in question. With a 0iD control, bit number one of parameter number one specifies the program format used with multiple repetitive cycles (one-line or two line style). You can set it in your initializing program to ensure that the control is appropriately set to accept programs utilizing your desired program format method.

Truly, any time you find yourself manipulating parameters in order to get programs to run properly is an example of when you should consider adding commands to your initializing program.

Here is an example for FANUC 0iD control that is being used on a turning center.

- O9010
- (Normal safety commands)
- G18 G20 G23 (XZ plane, Imperial measurement system, stored stroke limit off)
- G40 G67 G99 (Cancel tool nose radius compensation, cancel modal macro call, per revolution federate)
- (Parameter settings)
- G10 L52 (Select parameter entry mode)
- N0000 R110 (Inch, ISO, TV check off)
- N0001 R10 (Program format)
- N0020 R4 (Output device is memory card)
- N3410 R0 (Circular motion tolerance)
- N5130 R0 (Thread chamfering amount)
- N5133 R300 (Escape amount during multiple G71 and G72)
- G11 (Cancel parameter setting mode)
- M30

Again, this program will be executed when the control reads a G100 word (if the user defined G code parameter is set as discussed above). The first two commands probably resemble what you are currently doing in all of your CNC programs. They simply ensure that all initialized states are still in effect.

The G10 L52 command tells the control to instate the parameter entry mode. Each subsequent command specifies a parameter setting. The N word specifies the parameter number and the R word specifies the setting value. When finished, G11 cancels the parameter setting mode.

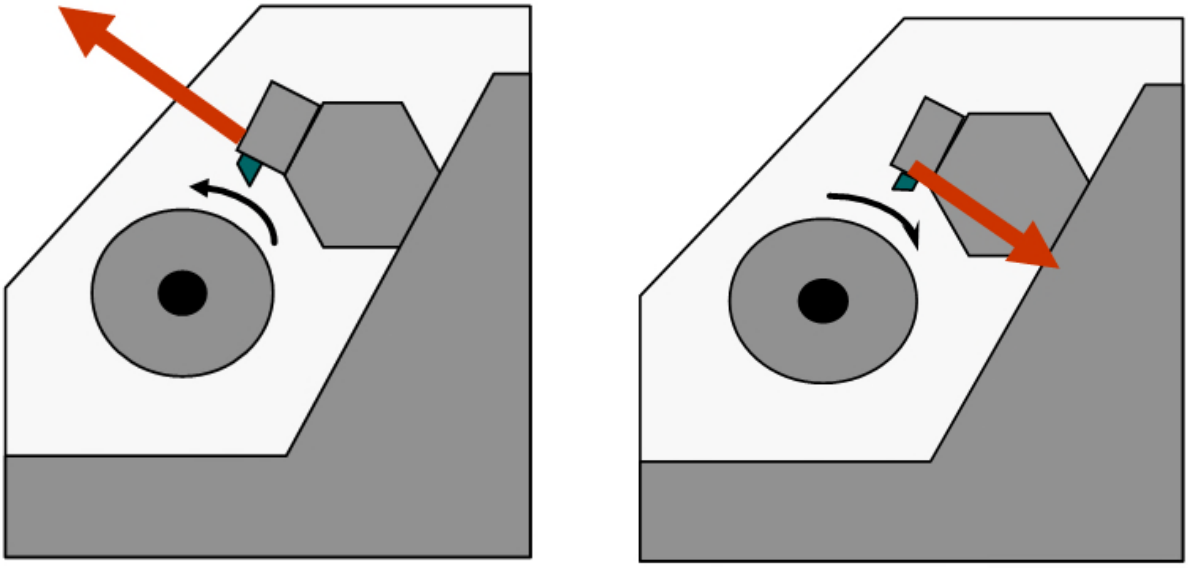
Note that you can even include parameter settings that may not be directly related to the way your programs run. In our example, we're setting the punch code format to ISO and turning off TV check. Standing for tape vertical check, if this setting is turned on, programs you load must include the same number of characters in every command (which is never the case). We're also setting the output device to the memory card.

Use the best “hand” of tooling when performing powerful machining operations on turning centers
Though this discussion is quite basic, many of the attendees in my CNC seminars are so surprised at this presentation, I can only assume that many CNC people have not considered the implications of this question.

Almost all current model slant bed turning centers allow CNC users to use either right and left hand tooling. With some machines, the hold down bolts are placed in different holes. On others, the cam-lock system is inverted. But by one means or another, each turret station can be made to hold tools of either type.

If holding tools designed to machine in a direction toward the work holding device (Z minus), right hand tools require an M03 (spindle forward) direction and left hand tools require an M04 (spindle reverse) direction. From a strictly cycle time perspective, changing spindle direction in the middle of the program based upon switching tool types (left to right hand and vise versa) takes time. For this reason, and since most cutting tools are more readily available in right hand versions (right hand tools are commonly less expensive and off-the-shelf items), and because some operations must be performed with right hand tools (tapping and right hand threading, for example), most CNC users prefer using right hand tools - and many use them exclusively.

Keep in mind, however, that your turning center’s rigidity and strength will commonly be better with left hand tooling. The drawing shows why. Notice that when left hand tools are used, the force of the machining operation is thrown into the machine’s bed and machining will be very stable. On the other hand, notice that when right hand tools are used, the force of the machining operation tends to pull the turret away from the machine bed, and machining is less stable.



For light duty machining, it truly doesn't matter which type of tooling you use. But as machining operations become more powerful (as is the case when performing heavy rough turning and boring operations), left hand tools are better. You would be best off using them and sacrificing the cycle time it takes for spindle reversals. And by the way, if you experience problems with inconsistent sizing (even for finishing operations) on tight tolerances, using left hand tools will improve the stability of machining.

Error trap your CNC programs

Mistakes made when running CNC machine tools will cause consequences. At the very least, the machine will not behave as expected. Worse, workpieces may be scrapped. And worse yet, damage may occur to the machine and/or operators may be injured. While nothing replaces skill and common sense when it comes to running CNC machine tools, CNC programmers should assume at least some of the responsibility for minimizing the problems caused when mistakes are made.

With parametric programming, a programmer can create tests within programs that can check and see whether a mistake has been made. This requires, of course, that the programmer is aware of the possibility that the mistake will occur (maybe it has occurred many times before) and has the ingenuity to come up with an appropriate way to test for the mistake condition.

One problem that programmers face when developing error-trapping routines is error-trapping commands must be placed in many CNC programs. This can be tedious and error prone. Additionally, you may be continually developing error-trapping techniques to handle new problems as they arise, and many commands will have to be added to all current programs. Many programmers give up on error trapping because it can be so tedious and time-consuming.

Rather than try to incorporate the specific error-trapping commands in every program, we recommend developing one universal error-trapping program that will be referenced from any program when testing for a problem. Only one command will be added to each program that tests for a given problem. With this technique, you will have but one program to maintain. If a test

condition changes, you will have but one set of commands to change. Additionally, you will be able to add new error-trapping techniques to this program at any time.

A pallet changer testing program

Many machining centers, especially horizontal machining centers, come with a two-pallet pallet changing system. The operator loads one pallet while the other is in cycle. It is quite important, of course, that when the operator activates the cycle, the correct pallet is in the machine. A mistake in this regard can be disastrous.

Unfortunately, most machine tool builders provide little in the way of pallet confirmation. Again, if the wrong pallet is in the machine, the control will simply activate the program, ignoring the problem. If your machine has parametric programming capabilities, it is quite easy to add your own pallet confirmation system. We're showing the example in Fanuc's custom macro B, but with a little ingenuity, you should be able to apply this technique with just about any version of parametric programming.

We'll use a #500 series permanent common variable with which to store the current pallet location. We'll make #501 = 1 if pallet A is in position and #501 = 2 if pallet B is in position. #501 is a permanent variable. Like an offset, its value will be retained even after the power is turned off.

With custom macro B, the function of your pallet change activation word (commonly an M60) can be changed to automatically update the value of #501 with each pallet change (see below). A test can be made at the beginning of each program to confirm that the correct pallet is in position before allowing machining to occur.

Note that this is a software solution to the problem and the control could get mixed up if the program is stopped in the middle of a pallet change. A more positive (but much more difficult) method would be to use the #1000 series input signal terminals in conjunction with physical limit switches mounted on the pallet changer to determine which pallet is in position.

Here is the "easy way", using #501 as a flag to determine which pallet is in position:

You must first manually set the value of #501 (just like you set an offset) to initially specify which pallet is in position. You'll find #501on the variables page of your display screen. If pallet A is currently in position, manually set #501 to 1.0. If pallet B is currently in position, set #501 to 2.0.

With custom macro B, a parameter must be set to make the control activate a specific program (O9001 in our case) whenever an M60 is activated. You must reference the custom macro descriptions in the programming manual for your specific control to confirm the parameter number. For a 16M Fanuc control, for example, it is parameter number 6071. You must set this parameter to a value that corresponds to your pallet changing M code. If your pallet changer is activated with an M60, set this parameter to a value of 60. From this point, whenever the control reads an M60, it will execute program O9001.

Next, you must load program O9001 into your control.

- O9001 (Pallet changing program - control will execute this program when M60 is read)

- M60 (Normal pallet change command)
- IF [#501 EQ 1] GOTO 1 (IF pallet A WAS in position GOTO N1)
- #501 = 1 (If pallet B WAS in position, set #501 to 1, pallet A now in position)
- GOTO 2 (Skip N1)
- N1 #501 = 2 (Set #501 to 2 - pallet B now in position)
- N2 M99 (Return to calling program)

Now, in every program that requires the correct pallet to be in position, add these commands at the very beginning:

If pallet A is supposed to be in position:

- O0001 (Pallet A is supposed to be in position right now!)
- IF [#501 EQ 1] GOTO 1
- #3000 = 100 (WRONG PALLET IN POSITION)
- N1 (Normal commands in program)
- .
- .
- .

- If pallet B is supposed to be in position:
- O0001 (Pallet B is supposed to be in position right now!)
- IF [#501 EQ 2] GOTO 1
- #3000 = 100 (WRONG PALLET IN POSITION)
- N1 (Normal commands in program)
- .
- .
- .

The #3000 system variable that will, if activated, put the machine into an alarm state (if all is okay, the control will skip this command). The message

MC-100 WRONG PALLET IN POSTION

will appear on the display screen.

An example:

The example error-trapping program we show allows the testing of three specific problems. Again, you can include as many test conditions as you need in this program. In test number one, we're testing to confirm that the machine is at its Y and Z axis zero return position (may be necessary on a horizontal machining center before a table index can safely occur). If not, an alarm will be sounded.

In test number two, we're testing that the tool length compensation value for a given tool is greater than the specified value (the operator has entered an appropriate value).

- O7001 (Universal error-trapping program)
- IF[#8 EQ 1] GOTO 100
- IF[#8 EQ 2] GOTO 200
- IF[#8 EQ 3] GOTO 300
- #3000 = 100 (NO ERROR SPECIFIED)
- **N100** (Begin handling first problem – Machine at Y and Z home)
- IF [#5022 EQ 0] GOTO 101 (If Y axis is at ZR position, skip alarm)
- #3000 = 101 (Y AXIS NOT AT HOME)
- IF [#5023 EQ 0] GOTO 199 (If Z axis is at ZR position, exit)
- #3000 = 102 (Z AXIS NOT AT HOME)
- N199 M99 (Return to calling program)
- **N200** (Begin handling second problem – Tool minimum length)
- IF [#20 NE #0] GOTO 201 (If T has been programmed, skip alarm)
- #3000 = 104 (T MISSING)
- N201 IF [#11 NE #0] GOTO 202 (If H has been programmed, skip alarm
- #3000 = 105 (H MISSING)
- N202 IF [#[2000 + #20] GE #11] GOTO 299
- #3000 = 106 (TOOL TOO SHORT)
- **N300** (Begin test for which pallet is in position)
- IF[#20 EQ 1.0]GOTO301
- IF[#20EQ2.0]GOTO 302
- #3000=105 (BAD PALLET SPECIFICATION)
- N301 IF [#501 EQ 1] GOTO 299
- #3000 = 100 (WRONG PALLET IN POSITION)
- N302 IF [#501 EQ 2] GOTO 299
- #3000 = 100 (WRONG PALLET IN POSITION)
- N299 M99

Again, you can include as many test conditions as you need. Note that we've begun each test condition with a new series of sequence numbers to avoid repeating them as new tests are added. We've also specified a new alarm number for each problem. This will allow you to come up with your own alarm list (like what's in the operator's manual) to further document each problem.

When you want to include a test of the Y and Z zero return position (probably before a table index), simply include the following command. Note that this test condition requires no input data to complete the test. We simply include the E word to specify which test to perform.

- N055 G65 P7001 E1.0 (Perform test number one)

When you want to test a tool offset value for minimum length, include the following command. Note that this test requires a tool station number (with T) and a minimum length (with H).

- N205 G65 P7001 E2.0 T6.0 H3.0 (Perform test number two)

When you want to test if the correct pallet is in position, give this command:

- N055 G65 P7001 E3.0 T1.0 (Perform test number three)

Letter address E specifies the number for the test to be performed (pallet check). The T-word specifies the pallet number that is supposed to be in position (1 or 2).

When to limit spindle speed on CNC turning centers

Constant surface speed is a great turning center feature. It simplifies programming because the speed is specified directly in surface feet per minute (meters per minute if you work in the metric mode). The control constantly and automatically selects the correct speed in rpm based on the tool's current diameter. Constant surface speed also ensures consistent finish as long as you program feedrate in per revolution fashion. And constant surface speed maximizes tool life, since the tool is always cutting at the appropriate speed.

For as good a feature that constant surface speed is, there are a couple of drawbacks. If it's not programmed wisely, for example, it can be a cycle time waster (we've addressed this issue in a past CNC Tech Talk column).

Another drawback that can have serious consequences has to do with your machines maximum spindle speed. As you know, the machine will automatically select the appropriate spindle speed in rpm based upon the programmed speed in surface feet per minute (or meters per minute) and the tool's current diameter. When you face a workpiece to center, the tool will go to zero diameter, and the spindle will accelerate to its maximum speed in the current spindle range. If the high range is currently selected, the spindle will run up to its maximum.

In most cases (round, true workpieces), this is exactly what is desired. Truly, the spindle speed is matched to the diameter being machined. But there are times when allowing the spindle to run up to its maximum could be disastrous.

Consider, for example a large, unbalanced, raw casting. Say the machine's maximum spindle speed is 6,000 rpm. If you face this casting to center in the program as one of the first machining operations, the spindle will run up to 6,000 rpm. And because the workpiece is unbalanced, it's likely that the workpiece will be thrown right out of the workholding device.

Another time when allowing the spindle to run up to its maximum will cause problems has t do with bar feeding applications. Many bar feeders can hold very long bars (as long as 12 or 15 feet). It's likely that if the bar is not perfectly straight and/or the bar feeder is not perfectly aligned, the bar feeder may not be able to keep up with the spindle. With your largest bar, for example, you may not be able to run the spindle faster than about 2,000 without serious vibration. (Note that many bar feeder manufacturers are overcoming this problem by providing bar feeders that hold and automatically load several shorter bars to achieve maximum spindle speed and still provide long periods of unattended operation. The only limitations with this style of bar feeder are that you must cut the bars prior to using them and you'll have more wasted stock in remnants.)

All turning center control manufacturers provide a way to specify a maximum spindle speed in the program. Most use a G50 or G92 word with which to specify maximum speed. For example, the command

- N005 G50 S1500

tells the control not to allow the spindle to run faster than 1,500 rpm, even if using constant surface speed and the current diameter is small enough to require a faster speed.

Unfortunately, determining the maximum speed for a given workpiece can be almost impossible to predict. In almost all cases, a test must be made during setup to determine this speed. And since this specification has so much to do with safety, if you do make a mistake, you must err on the side of caution.

In the case of the large, unbalanced casting, for example, the setup person must carefully test for maximum speed in rpm. After loading a casting, they'll start the spindle at a very slow speed in rpm. Slowly, and by small increments, they'll increase the speed until the machine begins to vibrate. Note that at this point they've already exceeded the maximum speed, so the actual maximum speed must be something slower (We recommend 20 percent slower). Since castings very, we recommend performing this test on a few workpieces to find the worst case scenario.

What some programmers forget, however, is that once the raw casting has been rough machined, it's likely that it will be running much more true than it does in its raw state. After roughing the first workpiece, the setup person can perform the maximum speed test again. While spindle speed will likely still have to be limited, the new maximum speed will probably be much faster than that for the raw casting. And since cycle time is related to spindle speed (time gets shorter as spindle speed is increased), you'll see an improvement to cycle time by incorporating a second spindle limiting command in the program. Note that the same is true in the bar feeding application. It's likely that as the bar gets shorter, the maximum spindle speed can be increased.

NEVER program around machine problems

The dwell command (G04 for most controls) will cause axis motion to pause for a specified period. The period is commonly specified in seconds. With one popular control the command

- G04 X1.0

Will cause the axes to pause for one second. Note that all other activities (spindle, coolant, etc.) will continue to function. Only the axes will pause.

The primary application for the dwell command is to allow time for tool pressure to be relieved. When grooving on a turning center, for example, most programmers like to include a short pause after the grooving tool has reached the groove bottom to allow the grooving tool to clean up the bottom of the groove. In like fashion, after plunging an end mill to the bottom of a pocket on a machining center, most programmers include a short pause to allow the tool pressure to be relieved.

We've seen some CNC users, very inappropriately, use the dwell command in order to *program around* certain machine limitations. (Worse, we've seen some machine tool builders actually recommend using the dwell command in this manner.)

Say, for example, your CNC turning center's coolant system isn't reacting fast enough. Maybe there's a bad check-valve. You specify an M08, but it takes 2-3 seconds before the coolant comes on at full blast. Yet you have a tool (possibly a coolant-through-the-tool drill) that requires coolant to be flowing at its maximum before the drill can enter a hole. If it's not, the drill could be damaged. In this case, some programmers will simply program an 2-3 second dwell command after the M08 and before the drill is allowed to enter the hole. While this does take care of the problem, it adds time to the cycle and who's to say that the coolant system's check valve won't eventually worsen (taking 5-6 seconds to fully activate the coolant). The right way to handle this problem is, of course, to *fix the machine*.

With the coolant system example, at least the machine was originally designed to function properly. It was a machine problem (the bad check valve) that caused the programmer to add the dwell. However, we have seen many poorly interfaced M codes (designed by the machine tool builder!) that do not provide full confirmation that the M code has been completed prior to allowing the machine to continue with the program.

One turning center manufacturer, for example, did not fully interface the chuck-jaw open and close M codes. When an end user attached a bar feeder, they found that when the jaw close M code was specified (to clamp on the bar after feeding), the program continued before the jaws fully closed. The bar stop moved away while the jaws were still closing and allowed the bar to feed too far. The proper long term solution to this problem would be, of course, to have the jaw closing M code fully interfaced so that the machine must receive a confirmation signal that the jaws are closed *before* it is allowed to continue with the program. Yet when this user contacted the machine tool builder, they were told to include a dwell command after the jaw close M code to allow time for the jaws to close. Programming around non-interfaced M codes with the dwell command can be very dangerous. In this example, if for any reason the jaws don't close in the allotted time, the program will simply continue.

You should *never* have to use the dwell command to program around non-interfaced M codes. Given the technology of today's programmable controller logic, machine tool builders can modify the way M codes behave with relative ease. So if you happen across an M code that is not fully interfaced, and especially if its function could be hazardous, we urge you to contact the machine tool builder to have it fully interfaced (and don't take no for an answer). CNC machines are dangerous enough when all M codes *are* properly interfaced.

Other M codes you should *never program around* with dwell commands include indexing devices, pallet changers, tailstocks, tool changing, or just about any other machine function that takes time to complete.

Another time we've seen the dwell command rather inappropriately applied has to do with spindle acceleration. With most turning centers, if the spindle is not yet up to speed when a positioning movement is completed, the control will make the machine wait until the spindle is up to speed. As

you *accelerate*, not waiting for the spindle may not be too dangerous (machining will simply be too slow). However, say you've just faced a workpiece to center using constant surface speed and the spindle is running at its maximum speed. Now you rapid this tool up to a six inch diameter to do some rough turning. Its likely that the (three inch) motion will occur faster than the spindle's deceleration to the appropriate speed for a six inch diameter.

In this case, if the machine does not wait for the spindle to slow down before the tool is allowed to continue, machining will occur *much* faster than it is supposed to. This could cause damage to the tool and/or workpiece and if the workpiece is not securely held, it could be thrown from the work holding device. This may be another time when you may be tempted to program a dwell command to handle a machine problem. You could, of course, include a dwell command right after the machine's movement to the six inch diameter that is long enough for the spindle to slow down to the appropriate speed. However, if you do this kind of work on a regular basis (several different workpieces), it's likely that you'll eventually forget to include the dwell command – and the results could be disastrous. Again, the better long-term solution is to get the machine tool builder to *fix the machine*. This problem may be as simple to solve as a parameter change.

NEVER let that bar hang out – and other safety issues

Certain machine shop practice rules must never be broken. If they are, the results can be disastrous. Experienced machinists will seldom break these rules. But we're seeing a large number of newcomers entering the manufacturing environment – people that have had little or no formal training in machine shop practices.

You must remember that newcomers to manufacturing are probably used to very safe working environments. Consider, for example, people entering the manufacturing field from high school. Their previous working experience may consist of working in local businesses. While mistakes they've made may have been costly, they probably haven't been in any real danger. They are now being placed in a much more dangerous working situation. If they make mistakes in a manufacturing company, it is possible that the result will be a personal injury. Here is one specific example.

Many CNC turning centers have a hole through the chuck and all the way through the spindle to accommodate bar feeders. A bar feeder will, of course, send the bar through this hole to provide the raw material to be machined. But if the machine does not have a bar feeder, many companies that have limited bar feed applications will use a *bar puller*. The puller is mounted to a turret station and draws the bar out at the completion of each workpiece. The bar (raw material) must be of an appropriate length prior to being put in the machine. *Under no circumstances, should the bar protrude from the rear end of the spindle* (out the back of the machine). If it does, it will be under severe centrifugal force as the spindle runs up to machining speed. At some point, the bar will radically bend to ninety degrees, shredding anything in its way (including the operator).

In one company I know of, CNC turning operators are responsible for cutting their own bars prior to loading them into the turning center. One new operator decided to minimize the number of times the bar was cut to reduce the amount of work he had to do. He was never made aware of how dangerous it would be to let the bar hang out the back end of the machine. Fortunately, when the

bar did bend to ninety degrees, this operator was not in close proximity and was not injured. However, the machine's spindle motor was not so lucky.

We sometimes forget how little entry-level people know. Most entry-level people will not even realize that a greater potential for danger even exists in their new job unless you point it out. Most will blindly follow the instructions they've been given. If something out of the ordinary happens – something you haven't warned them about – they won't understand the consequences of making a mistake. And most will simply proceed.

In many cases, we're talking about very basic mistakes. Mistakes that no *well trained* CNC person would ever make. What happens when you load a tool into the wrong tool changer position? What if you start the program with the wrong side of a pallet facing the spindle? What if you forget to tighten work-holding clamps? What if you run a tool past the point when it gets dull? (How do you tell *when* a tool is dull?) These are all mistakes that will result in scrap workpieces, damaged tooling and machines, and possibly injury to the operator.

Consider the kind of training you currently supply entry-level people. Have you had some close calls? If you haven't had any catastrophic mistakes, is it because you have worked to make each operator's job fail safe? Or is it because you've just been lucky?

Admittedly, people running CNC machines must take on some of the responsibility for their work. In days gone by, people entering the field of manufacturing came from technical schools and apprenticeship programs. They had a firm understanding of machine shop practices before entering the shop. But as we employ people with lesser and lesser previous machine shop experience, we must also take on more of the responsibility for providing safe work environments.

This will require a combination of efforts. First, we must engineer our processes in a more fail-safe manner. Consider potentially catastrophic mistakes a newcomer can make and eliminate the potential for making them. In the case of the bar puller example, this may mean keeping operators from cutting their own bars.

Second, we must better educate new people. Make them aware of potentially dangerous situations. Ensure that they sure they understand that if something out of the ordinary occurs, notify the person in charge (they shouldn't simply proceed).

Third, we may have to better compromise manufacturing aggressiveness with safety. Experienced people will have the ability to keep up with aggressive processes (loading workpieces, measuring workpieces, performing secondary operations, etc.). But newcomers may not have the needed skill and won't be able to keep up. In their efforts to keep up, they will be prone to making mistakes.

Efficiency

Finally, you want your CNC programs to execute as quickly as possible. While many things can be done with process to minimize cycle time, we concentrate on CNC programming techniques.

Structure your programs to minimize program execution time

A properly formatted program will execute faster than a poorly formatted one. There are many structure-related techniques that cost you nothing to implement. Here we show a few.

Efficient tool changing / turret indexing

Machine tool builders have come up with some pretty unique devices with which to automatically change tools on machining centers. Though this is the case, the programming words related to tool changing are relatively consistent from one machining center to the next.

Machining center suggestions

The T word is commonly used to rotate the machine's magazine (or tool storage device) to bring a tool into the *ready* or *waiting* position. An M06 exchanges the tool in the spindle with the tool in the ready position (though on some machines, the T word also does this). And an M19 rotates the spindle to its orient position (to align the key in the tool changer arm with the keyway in the tool holder). Though most automatic tool changing devices are easy to program, there are some techniques you can use that will minimize tool changing time.

Where is the tool change position?

First of all, machine tool builders vary when it comes to where you must send the machine prior to commanding a tool change. Some (very few) do *not* require a special positioning movement. They can change tools anywhere as long as the tool change will clear the work holding setup. If you are lucky enough to have this kind of automatic tool changer, you can dramatically minimize tool changing time by keeping the tool change position close to the workpiece being machined. This minimizes the amount of rapid movements required to get to the tool change position.

Most machining center builders require that you send the machine to its reference position (also called home position, zero return position, and grid zero position) in at least one axis in order to line up the tool changer mechanism with the tool in the spindle. With this kind of machine, be sure to limit the number of moving axes to those required. Most vertical machining centers, for example, require that you send only the Z axis to its reference position. Additionally sending X and Y to their reference position would be wasteful (unless there is some obstruction that would cause interference during the tool change).

Get the next tool ready

Many (though not all) machining centers allow you to be rotating the magazine to get the next tool ready while the tool in the spindle is machining the workpiece. This saves magazine rotation time, since the next tool will likely be ready when the spindle tool is finished. With most machines, this simply involves including a T word in the program after each tool change to specify the tool number for the next tool.

Orient the spindle on the tool's retract to the tool change position

As stated, M06 is the word that causes the tool change. It will perform all motions (including spindle orientation) related to tool changing. However, spindle orientation takes time (usually 1-3 seconds), and waiting for the machine to retract to its tool change position *before* the spindle begins orienting

is wasteful. You can include an M19 with the machine's motion to the tool change position, which will save 1-3 seconds per tool change.

Load tools sequentially for short cutting operations

All current model tool changers are *random access*. You can select any tool at any time within the CNC program. While this is a great feature, there are times when selecting tools sequentially (tools next to each other in the magazine) will minimize cycle time. If for example, you have center drill that machines but one hole in aluminum, it is likely that it will finish its operation and the machine will be back at the tool change position *before* the magazine can rotate to the next tool (especially if the magazine must rotate a long distance to get the next tool ready).

By the way, this can also cause inconsistency in cycle time from one time a job is run to the next. One time tools are placed into the magazine in an efficient manner (right next to each other). The next time the job is run tools may be less efficiently placed in the magazine. This is one good reason to document cycle time right in the CNC program (with a message). The setup person can easily time the current cycle to confirm that it is running as efficiently as it should.

Look for machines that allow tool replacement during automatic operation

Note that many machine tool builders completely lock-out operation of the tool changer magazine during automatic operation. While this is done for obvious safety reasons, there are times that having access to the magazine for tool replacement can increase productivity. If for example, a tool is dull, or is the setup person would like to start loading tools for the next job, doing so during automatic operation can minimize machine downtime. More and more machine tool builders are providing an interlock for the tool changer magazine that allows safe manual operation of the tool changer magazine even during the machine's automatic operation.

Turning center suggestions

The cutting tools in your turning center/s change on a regular basis. To minimize tool changing time during setups, many setup people will simply load the tools required for the new job and leave tools in the turret from the last job as long as they don't interfere with the new job. This practice is very common, especially when users want to incorporate standard tool stations for their most commonly used tools.

The tool change position must, of course, be far enough away from the workpiece/chuck to allow indexing with all tools *currently* in the turret. A long boring bar or drill left in the turret from the last job could cause real problems! To handle this potential problem, many users simply make their turret index position far enough away from the chuck to allow all tools they own to safely clear when indexing. Some users make the machine's zero return (reference) position the index position for this very reason. At the zero return position, the turret of most turning centers can safely index any tool into position without interference.

Many (indeed most) jobs would allow the turret to safely index much closer to the chuck, meaning it isn't always necessary to go all the way back to the zero return position to change tools. A job with all

turning tools (no drills or boring bars currently in the turret) will allow the turret index position to be quite close to the chuck/workpiece. However, if there are some boring bars required in the job (or if they're left in the turret from the last job) the turret index position must be further away.

With large production quantities, the best way to handle this problem is (of course) to remove all unneeded tools from the turret during setup, and to make the turret index position as close to the workpiece as is safe to do. But with lower quantities or any time reducing *setup* time is quite important, wouldn't it be nice if you could quickly select a safe index position during setup that would be both safe and efficient for the *current* turret setup? With custom macro, and with a small programming format change, it's actually quite easy to do.

With the new setup made, and with all tools in the turret needed for the up-coming job, the setup person will (manually) move the turret to a save and efficient index position. With the machine resting at this safe index point, they run this program.

- O9000 (Index position memorizing program)
- #511 = #5021 (Memorize current X position relative to zero return position)
- #512 = #5022 (Memorize current Z position relative to zero return)
- M30 (if activated by automatic mode or M99 if activated by mdi)

All this program does is memorize the machines current position for use as the index position during the upcoming production run. #5021 and #5022 are system variables that constantly contain the machine’s position in X and Z relative to the zero return. The current machine position is being stored in permanent common variables #511 and #512. These values will be referenced in the cutting program during the production run whenever a tool change is required.

To eliminate having to keep referencing these system variables in the cutting program, we recommend using a short custom macro program that can be easily executed from the cutting program. While there are several ways of doing this (using a user defined G code, for example), here's an example that's pretty easy to program and understand.

- O0001 (Cutting program)
- N005 T0101 M42 (First tool)
- .
- .
- .
- .
- (Turret index required now)
- G65 P9021 (Move to previously selected tool change position)
- T0202 (Second tool)
- .
- .
- .
- (Tool change required now)
- G65 P9021 (Move to previously selected tool change position)

- T0303 (Third tool)
- .
- .
- .

Here's the short custom macro O9021.

- O9021 (Move to safe index position)
- G00 U[#511 - #5021] W[#512 - #5022] (Incrementally, rapid to save index position)
- M99 (End of custom macro)

Program Constant Surface Speed Efficiently

Constant surface speed (usually specified with a G96), is the turning center feature that allows the control to constantly update the spindle speed in RPM based on a speed specified in surface feet per minute (or meters per minute) and the diameter the tool is currently cutting. It is a very helpful programming feature that makes programming simpler, enhances workpiece quality, and improves tool life. It is used for single point cutting tools (like turning tools, boring bars, and grooving tools).

While constant surface speed is an important programming feature that almost all CNC users should use, it can also be one of the worst cycle time wasters if not properly programmed. In this short article, we'll show the most efficient manner of programming constant surface speed.

To get an idea of why constant surface speed can waste cycle time, consider this example in which we will intentionally exaggerate the potential for problems. Say your tool change position for a given program is at about ten inches in diameter. Your program is written to machine a rather small workpiece with a rough diameter of 0.500 inch.

With the turret resting at the tool change position (10 inches in diameter), you give the command G96 S600 M03 to turn the spindle on at 600 surface feet per minute (SFM). At a ten inch diameter, this equates to 229 RPM. The spindle starts. Next you give the command to rapid the tool to the 0.50 diameter (G00 X.5 Z.1). During this command the spindle will increase to about 4,600 RPM, based on the 0.5 diameter. For most machines, the rapid motion command (only 4.75 inches in length) will occur much faster than the spindle increase from 229 RPM to 4,600 RPM. (It is not uncommon to take five to ten seconds for such an RPM change to take place.) At the end of the rapid movement, the machine will wait until the spindle increases.

For this first spindle start up, about the best you can do to minimize cycle time is include the spindle start command (M03) together with the motion command (the G00 command). This will make the control start the spindle at the same time movement begins. In this example, at least the rapid positioning movement will be internal to the spindle start (saving about one second of cycle time if the machine has a rapid rate of 400 IPM).

At the completion of the first tool, you send the machine the machine back to the ten inch tool change position for the tool change. During this motion the spindle will slow to 229 RPM, wasting another 5 to 10 seconds! As the next tool approaches and the spindle increases, yet more cycle time is wasted.

All of this is not to mention the wear and tear on your spindle drive system as the spindle drastically changes RPM throughout your program. Also, electricity is being wasted since it takes additional power to accelerate and decelerate the spindle.

The key to programming constant surface speed efficiently is keeping the spindle from changing RPM during tool changes. This can be easily accomplished by temporarily switching to RPM mode (G97 on most controls) while tool changes are made. For each tool that uses constant surface speed, on its return to the tool change position, temporarily switch to RPM mode. Include in this command the RPM for the next tool's first approach movement. This will keep the machine from making drastic RPM changes and effectively reduce cycle time. Here is a simple program for the previous example that stresses this technique.

- O0001 (Program number)
- N005 T0101 (Index to tool one)
- N010 G00 G96 X0.5 Z0.005 S600 M03 (Rapid into position, start spindle on the
- way)
- N015 G01 X-0.05 F0.005 (Rough face)
- N020 G00 Z.1 (Rapid away)
- N025 G00 G97 X10. Z5. S5348 (Rapid to tool change position, select RPM for the
- next tool's first approach position)
- N030 M01 (Optional stop)
- N035 T0202 (Index to tool two)
- N040 G00 G97 X.5 Z0 S700 M03 (Rapid into position, spindle commands are only
- required if restarting tool)
- N045 G01 X-.06 F.004 (Finish face)
- N050 G00 Z.1 (Rapid away)
- N060 G00 X10. Z5. M05 (Rapid to tool change position, stop spindle)
- N065 M30 (End of program)

First, notice we start the spindle in the first tool's approach to make the approach internal to the spindle start (line N010). Next, notice that on the first tool's return for tool change (line N025), we switch to RPM mode and increase to the second tool's beginning RPM (calculated by multiplying 3.82 times 700 SFM and divided the result by the 0.50 approach diameter). In line N040, we include the spindle command just in case the operator wishes to rerun tool number two. Finally, notice in line N060, we turn off the spindle during the last tool's return to the tool change position. This makes the return movement internal to the spindle stop.

Program spindle range changes appropriately

Metal cutting CNC machines have spindles. The spindle rotates the cutting tool on CNC mills. With lathes, it rotates the workpiece. You probably know that larger CNC mills and lathes have multiple spindle ranges. Low ranges provide more power while high ranges provide more speed. Ensuring that

machining is done in the appropriate spindle range is of critical importance to achieving optimum machining efficiency.

1) Know your spindle's power and speed characteristics

Machine builders publish spindle characteristics in their operator manuals. Here you will find the lowest and highest rotation speed (revolutions per minute) for each range, along with how much power can be expected throughout its rpm spectrum.

If you have never studied this important data, it is likely that your cycle times are longer than they need to be. Worse, you may be placing undue stress on – or even stalled – a machine's spindle motor. So, dig out the manual and learn about spindle characteristics.

2) Know how long it takes to change spindle ranges

There are at least two kinds of spindle-range changing systems:

1) Those having spindle drive motors that incorporate multiple windings – Changing ranges occurs electronically by changing which motor windings are used. Range changing occurs nearly instantaneously.
2) Those with mechanical transmissions – The highest range may be direct drive. Lower ranges engage gears within the transmission. Range changing may take several seconds, especially if the spindle must stop during the process.

3) Know how to select spindle range

With machining centers, spindle range changing is somewhat transparent. Since spindle speed is specified in rpm, the S-word that specifies speed also causes the machine to select the related spindle range. Say a machine's low range runs from 20-1,500-rpm. The high range runs from 1,501-4,000-rpm. An S-word of S300 causes the machine to select the low range. An S-word of S2000 causes the machine to select the high range.

An ill-informed programmer can unwittingly cause two problems:

1) The program may be causing unnecessary range changes from tool to tool. With mechanical transmissions, this will increase cycle time, but may go unnoticed since it only manifests as some tools taking longer to change than others. Running tools requiring the same range in sequence will reduce cycle time.
2) The spindle speed rpm calculation for a powerful roughing operation may put the spindle at the low end of the high spindle range, where there is limited power. This will place undue strain on the spindle drive system or cause the spindle motor to stall. A well-informed programmer will compromise spindle speed slightly downward, selecting the highest speed in the low range – where there is ample power to perform the machining operation.

With turning centers, spindle range changing is done with M-codes – and higher ranges usually overlap lower ranges. For a turning center with three spindle ranges, the low range may be selected with M41 and run from 30-1,400-rpm. The middle range with M42 and runs from 40-2,800-rpm. And the high range with M43 and runs from 45-4,500-rpm.

4) Know how spindle speed can affect cycle time

This point applies only to turning centers – and operations that utilize constant surface speed. With constant surface speed, the CNC will constantly select speed in rpm based on a speed specified in surface feet or meters per minute and the diameter currently being machined. Our example scenarios use spindle characteristics just described.

Spindle speed is inversely proportional to time – assuming feedrate is programmed in per-revolution fashion. If you can double spindle speed, the time required for the related machining operation/s will be cut in half.

A popular rule of thumb for spindle range selection is to rough machine in the low range and finish machine in the high range. While this is good rule of thumb for ensuring that the spindle has adequate power, it does not consider speed so well.

Consider a 1.0-in diameter workpiece that must be rough- and finish-turned. The recommended speed is 500-sfm for the roughing tool. Even at the largest diameter (1.0-in) this renders 1,910-rpm (3.82 times 500 divided by 1.0). Smaller diameters will require even higher rpms. If the programmer selects the low range based on the rule of thumb, the spindle will be limited to 1,400-rpm. The roughing operation will be done faster in a higher range, assuming there is adequate power.

5) Know when to change spindle ranges

This point also applies only to turning centers and roughing operations requiring constant surface speed. Consider rough turning a 4-in diameter shaft having several diameters, the smallest being a 1.0-inch diameter. Say the recommended speed is 800-sfm. At 4.0-inches, the required speed is 764-rpm. And of course, the low range will be used to provide the needed power.

As rough turning continues, diameters get smaller and rpm increases. At 2.125-in, over 1,400-rpm is required, but if machining continues in the low range the spindle will peak out at 1,400-rpm – and each successive roughing pass will take longer than it should. It would be wiser to switch to the middle range at this point, especially if range changing is instantaneous.

Reduce air cutting time when appropriate (rapid approach distance and feed off amount)

You know that a certain amount of cutting motion time is *air cutting* time. Air cutting time occurs when the cutting tool moves from its approach position until it contacts the workpiece. It is also time when the tool feeds off a surface until it reaches a safe clearance position.

Most programmers use a standard approach and feed-off distance of 0.100 inch (or 2.5 mm. Consider center drilling fifty holes on a machining center at 5.0 inches per minute. If using a rapid approach distance of 0.1 inch, there will be 5.0 inches of air cutting motion in the program for this tool (50 holes times 0.1 inch). At a feedrate of 5.0 inches per minute, this equates to one minute of air cutting time. From this point, of course, the holes must be drilled. And after that, there may even be more operations on these fifty holes (counter boring, tapping, reaming, etc.). If each of these tools use a 0.100 inch rapid approach distance, of course, at least five more inches of air cutting time will occur per tool.

While 0.100 inch is a very safe rapid approach distance, you must consider the impact it can have on air cutting time. There are times when 0.100 inch may be an excessive rapid approach distance, at least when the surface the tool is approaching is qualified and when the tool being used has been accurately set. By qualified, I mean that the surface is not varying by very much. Possibly the surface has been machined (varying by less than a few thousandths of an inch). Or possibly the surface has been cold drawn (varying by 0.005 to 0.010 inch). Possibly the tool has just machined one surface and is about to machine another.

In these cases, you can safely reduce rapid approach distance. Make your rapid approach distance about ten times the total of the surface variation amount plus the total possible imperfection in tool setting. If the surface is varying about 0.003 inch and the cutting tool setting could vary by as much as 0.002 inch, for example, a 0.050 inch rapid approach distance should be sufficient.

In the case of the fifty center-drilled holes, this would allow us to cut the rapid approach distance and air cutting time in half. Instead of taking one minute, air cutting time will take only thirty seconds (based on the 5.0 ipm feedrate mentioned above).

We restate two safety-related points about applying this technique. First, the surface must be qualified. Don't try to reduce rapid approach distance for sand castings, forgings, and other workpiece blanks that vary from workpiece to workpiece or from lot to lot. In these cases a rapid approach distance of 0.25 inch or more may be necessary.

Second, your tool setting positions must be accurate. For machining centers and when approaching in the Z axis (as when approaching with hole-machining tools), this means your tool length compensation values must be accurately measured and entered. When approaching in XY (as when approaching with an end mill), the cutter cannot be larger in diameter than expected. For turning centers, this means that program zero assignments for each tool must be correctly measured and entered.

Assuming you have already been setting tools pretty accurately, I contend that if you would have started your CNC career using a 0.050 inch rapid approach distance for qualified surfaces, you would have had no more problems approaching with cutting tools that you haven't already had using the 0.100 inch rapid approach amount. You still, of course, have the same program verification functions that you currently use whenever approaching with a new tool. These functions include dry run, single block, rapid override, distance-to-go, and feed hold. If you've had problems with crashes, they've probably not been related to the size of your approach distance.

These points also apply when you feed a tool away from a surface after cutting. If anything, it will be safer to reduce feed-off distance, since the tool is not moving at rapid prior to this motion – and during this motion the tool is still at its cutting feedrate.

One perfect example of when you can safely reduce rapid approach distance on turning centers is when you rough face and rough turn with the same tool. Possibly you rapid the face-and-turn tool within 0.100 inch of the diameter to be faced (on the side). You then face the end of the workpiece to center. Next you retract the tool 0.100 away from the workpiece in Z (still in G01?). You then rapid the tool straight to the first diameter it will rough turn, maintaining the 0.100 clearance.

Finally, you program the tool to rough turn its first diameter. In this example, the very tool that rough faced the workpiece is being used to rough turn the workpiece. The variation in the surface being approached (workpiece face) will be next to nothing. Maintaining the 0.100 approach distance for the rough turning pass/es will be very wasteful.

Check parameters that affect cycle time

We already mentioned that parameters control the impact certain machine functions (like spindle acceleration and deceleration) can have on cycle time. There are others you should be aware of. At the very least, you must recognize the potential that a parameter may be affecting cycle time. If a machine pauses during program execution for no apparent reason, you must determine the reason why. It is possible that an inappropriate parameter setting is causing the pause. Here are three examples related to canned cycles (machining centers) and multiple repetitive cycles (turning centers)

G73 chip break peck drilling cycle

As you probably know, G73 causes the drill to peck into the hole a specified amount (the Q value), and then retract a small amount to break the chip. This retract amount is set by parameter and should be but a tiny amount (usually 0.002 to 0.005 inch will suffice). Too excessive a retract amount will waste program execution time because the very next motion will be at a feedrate to the next peck amount. We have seen machine tool builders that excessively set this parameter to 0.100 inch. With a 0.100 peck amount, this actually doubles the time it will take to peck drill the hole!

G83 deep hole peck drilling cycle

Like G73, G83 is used for peck drilling. But G83 is used to clear chips between pecks. G83 will cause the tool to feed into the hole by the peck amount, retract all the way out of the hole, and then rapid back into the hole to within a small amount from where it left off. This small amount is set by parameter. Depending upon drill size and workpiece material, should be set from about 0.020 to 0.050 inch. The larger the hole and the more brittle the material, the more likely chips will fall back into the hole during retract.

G71 rough turning cycle

While the point made here is shown for G71, it also applies to G72 (the rough facing cycle). G71 causes the tool to make a series of rough turning passes. For each pass, the tool will rapid to its cutting diameter in X, feed to within a small distance of the next face in Z, and then feed off the workpiece a small amount in X and Z. This feed-off amount is set by parameter, and should be set to a tiny amount. Since the next motion is a rapid motion back to the starting point in Z, we recommend that this parameter be set to about 0.010 inch.

Where do you find parameter documentation?

If you're questioning how these parameters are set for your particular machines, or if you suspect some other time related parameter may be improperly set, look in the Fanuc Operator's Manual in the section that describes the feature in question. In the case of G83, for example, you can find the parameter number related to the approach distance after each peck in the series of notes that

accompany the G83 description. You'll have to go to the machine, of course, to find the actual value of this parameter setting.