

Programming two axes CNC lathe

Digital book

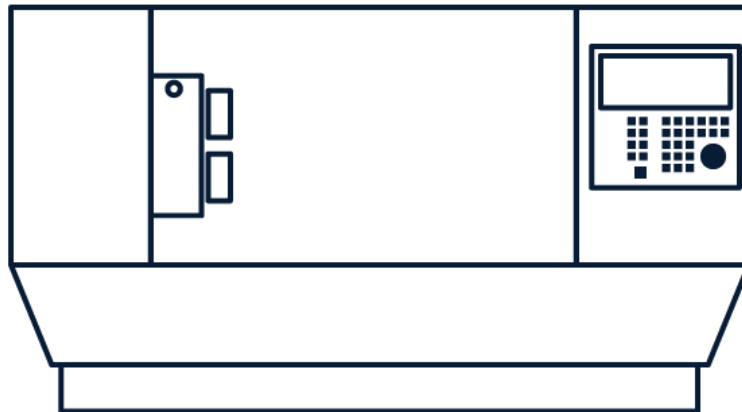


TABLE OF CONTENTS

1.	NUMERICAL CONTROL	1
A.	DEFINITION OF NUMERICAL CONTROL	1
B.	CONVENTIONAL AND CNC MACHINING	2
C.	NUMERICAL CONTROL ADVANTAGES	3
D.	TYPES OF CNC MACHINE TOOLS.....	5
E.	LATHES AND TURNING CENTERS	5
F.	PERSONNEL FOR CNC	6
G.	SAFETY RELATED TO CNC WORK.....	7
2.	COORDINATE GEOMETRY	9
A.	RECTANGULAR COORDINATE SYSTEM	9
B.	MACHINE GEOMETRY	13
3.	CONTROL SYSTEM.....	15
A.	GENERAL DESCRIPTION	17
B.	SYSTEM FEATURES	19
C.	MANUAL PROGRAM INTERRUPTION	23
D.	MANUAL DATA INPUT - MDI	24
E.	PROGRAM DATA OVERRIDE	24
F.	SYSTEM OPTIONS	29
4.	CNC TURNING	32
A.	TYPES OF CNC LATHES.....	32
B.	NUMBER OF AXES.....	32
C.	AXES DESIGNATION	33
D.	FEATURES AND SPECIFICATIONS	36
5.	PROGRAM PLANNING	39
A.	STEPS IN PROGRAM PLANNING.....	39
B.	INITIAL INFORMATION	39
C.	CNC MACHINE FEATURES	40
D.	PART COMPLEXITY	40
E.	MANUAL PROGRAMMING.....	41
F.	CAD/CAM AND CNC	41
G.	TYPICAL PROGRAMMING PROCEDURE.....	42
H.	PART DRAWING	43
I.	METHODS SHEET	49
J.	MATERIAL SPECIFICATIONS	49
K.	TOOLING SELECTION	51
L.	PART SETUP	52
M.	TECHNOLOGICAL DECISIONS	52
N.	WORK SKETCH AND CALCULATIONS.....	55
O.	QUALITY IN CNC PROGRAMMING	56
6.	PART PROGRAM STRUCTURE	58
A.	BASIC PROGRAMMING TERMS	58
B.	PROGRAMMING FORMATS	59
C.	WORD ADDRESS FORMAT	60

D.	FORMAT NOTATION	61
E.	SYMBOLS IN CNC PROGRAMMING	65
F.	TYPICAL PROGRAM STRUCTURE.....	66
G.	PROGRAM HEADER	69
7.	PREPARATORY COMMANDS	71
A.	DESCRIPTION AND PURPOSE	71
B.	APPLICATIONS FOR TURNING	72
C.	G-CODES IN A PROGRAM BLOCK.....	72
D.	GROUPING OF COMMANDS	75
E.	G-CODE TYPES	76
8.	MISCELLANEOUS FUNCTIONS	78
A.	DESCRIPTION AND PURPOSE	78
B.	TYPICAL APPLICATIONS	79
C.	M-FUNCTIONS IN A BLOCK.....	81
D.	PROGRAM FUNCTIONS	83
E.	MACHINE FUNCTIONS	86
9.	SEQUENCE BLOCK	91
A.	BLOCK STRUCTURE.....	91
B.	PROGRAM IDENTIFICATION	92
C.	SEQUENCE NUMBERS	93
D.	END OF BLOCK CHARACTER.....	96
E.	STARTUP BLOCK OR SAFE BLOCK	96
F.	PROGRAM COMMENTS	98
G.	CONFLICTING WORDS IN A BLOCK.....	98
H.	MODAL PROGRAMMING VALUES.....	100
I.	EXECUTION PRIORITY	101
10.	INPUT OF DIMENSIONS	103
A.	IMPERIAL AND METRIC UNITS	103
B.	ABSOLUTE AND INCREMENTAL MODES.....	105
C.	DIAMETER PROGRAMMING.....	109
D.	MINIMUM MOTION INCREMENT	109
E.	FORMAT OF DIMENSIONAL INPUT	110
F.	CALCULATOR TYPE INPUT	114
11.	SPINDLE CONTROL	116
A.	SPINDLE FUNCTION.....	116
B.	M03 AND M04. DIRECTION OF SPINDLE ROTATION	117
C.	SPINDLE STOP	120
D.	SPINDLE ORIENTATION	120
E.	SPINDLE SPEED - R/MIN	122
F.	SPINDLE SPEED - SURFACE.....	122
G.	CONSTANT SURFACE SPEED	124
12.	FEED RATE CONTROL	131
A.	FEED RATE CONTROL	131
B.	FEED RATE FUNCTION.....	131
C.	FEED RATE SELECTION	132

D.	ACCELERATION AND DECELERATION	133
E.	CONSTANT FEED RATE	136
F.	MAXIMUM FEED RATE	137
G.	FEED HOLD AND OVERRIDE.....	138
H.	E ADDRESS IN THREADING	140
13.	TOOL FUNCTION.....	141
A.	T-FUNCTION	141
B.	TOOL OFFSET REGISTERS	142
14.	REFERENCE POINTS.....	148
A.	REFERENCE POINT GROUPS	149
B.	MACHINE REFERENCE POINT	149
C.	PART REFERENCE POINT.....	151
D.	TOOL REFERENCE POINT	154
15.	REGISTER COMMANDS	155
A.	POSITION REGISTER COMMAND	155
B.	LATHE APPLICATION	156
16.	POSITION COMPENSATION	162
A.	DESCRIPTION	162
17.	WORK OFFSETS.....	168
A.	WORK AREAS AVAILABLE	168
B.	LATHE APPLICATIONS	170
C.	TOOL SETUP	172
18.	RAPID POSITIONING	176
A.	RAPID TRAVERSE MOTION	176
B.	RAPID MOTION TOOLPATH	177
C.	TYPE OF MOTION & TIME COMPARISON.....	182
D.	REDUCTION OF RAPID MOTION RATE	182
E.	RAPID MOTION FORMULAS	183
F.	APPROACH TO THE PART	183
19.	MACHINE ZERO RETURN	186
A.	MACHINE REFERENCE POSITION	186
B.	RETURN TO PRIMARY MACHINE ZERO	188
C.	RETURN POSITION CHECK COMMAND.....	192
D.	RETURN FROM MACHINE ZERO POINT.....	193
E.	RETURN TO SECONDARY MACHINE ZERO	195
20.	LINEAR INTERPOLATION	196
A.	START AND END OF THE LINEAR MOTION.....	196
B.	SINGLE AXIS LINEAR INTERPOLATION	196
C.	TWO AXES LINEAR INTERPOLATION.....	197
D.	THREE AXIS LINEAR INTERPOLATION	198
E.	PROGRAMMING FORMAT	198
F.	LINEAR FEED RATE	198
G.	PROGRAMMING EXAMPLE	201
21.	BLOCK SKIP FUNCTION	203

A.	TYPICAL APPLICATIONS	203
B.	BLOCK SKIP SYMBOL.....	203
C.	CONTROL UNIT SETTING	204
D.	BLOCK SKIP AND MODAL COMMANDS.....	204
E.	PROGRAMMING EXAMPLES	206
22.	DWELL COMMAND	219
A.	PROGRAMMING APPLICATIONS.....	219
B.	DWELL COMMAND	219
C.	DWELL TIME SELECTION	221
D.	SETTING MODE AND DWELL.....	222
E.	MINIMUM DWELL	222
F.	NUMBER OF REVOLUTIONS	223
G.	LONG DWELL TIME.....	225
H.	FIXED CYCLES AND DWELL.....	227
23.	MACHINING HOLES	229
A.	TOOL APPROACH MOTION.....	229
B.	TOOL RETURN MOTION.....	230
C.	PECK DRILLING CYCLE - G74	231
D.	TAPPING ON LATHES.....	233
E.	OTHER OPERATIONS	235
24.	CIRCULARINTERPOLATION.....	237
A.	ELEMENTS OF A CIRCLE	237
B.	QUADRANTS.....	238
C.	PROGRAMMING FORMAT	240
D.	RADIUS PROGRAMMING	244
E.	ARC PROGRAMMING.....	245
25.	TOOL NOSE RADIUS OFFSET	247
A.	TOOL NOSE TIP	247
B.	RADIUS OFFSET COMMANDS.....	247
C.	TOOL TIP ORIENTATION	248
D.	EFFECT OF TOOL NOSE RADIUS OFFSET.....	249
E.	SAMPLE PROGRAM.....	250
F.	MINIMUM CLEARANCE REQUIRED	252
G.	RETRACTION FROM A BORED HOLE	253
H.	CHANGE OF MOTION DIRECTION	254
26.	TURNING AND BORING.....	256
A.	TOOL FUNCTION - TURNING	256
B.	LATHE OFFSETS.....	257
C.	MULTIPLE OFFSETS	259
D.	OFFSET SETTING	265
E.	FUNCTIONS FOR GEAR RANGES.....	265
F.	AUTOMATIC CORNER BREAK.....	266
G.	ROUGH AND FINISHED SHAPE.....	272
H.	PROGRAMMING A RECESS.....	276
I.	SPINDLE SPEED IN CSS MODE.....	278

J.	LATHE PROGRAM FORMAT	279
27.	LATHE CYCLES	281
A.	STOCK REMOVAL ON LATHES.....	281
B.	PRINCIPLES OF LATHE CYCLES.....	281
C.	G90 - STRAIGHT CUTTING CYCLE	282
D.	G94 - FACE CUTTING CYCLE	290
E.	MULTIPLE REPETITIVE CYCLES	291
F.	CONTOUR CUTTING CYCLES	292
G.	TYPE I AND TYPE II CYCLES.....	294
H.	G71 - STOCK REMOVAL IN TURNING	296
I.	G72 - STOCK REMOVAL IN FACING	299
J.	G73 - PATTERN REPEATING CYCLE	302
K.	G70 - CONTOUR FINISHING CYCLE	305
L.	BASIC RULES FOR G70-G73 CYCLES	307
M.	G74 - PECK DRILLING CYCLE	307
N.	G75 - GROOVE CUTTING CYCLE	309
O.	BASIC RULES FOR G74 AND G75 CYCLES.....	310
28.	GROOVING ON LATHES	311
A.	GROOVING OPERATIONS	311
B.	GROOVE SHAPE	311
C.	GROOVE LOCATION	312
D.	GROOVE DIMENSIONS	313
E.	SIMPLE GROOVE PROGRAMMING	314
F.	PRECISION GROOVING TECHNIQUES.....	316
G.	MULTIPLE GROOVES.....	323
H.	FACE GROOVES	323
I.	CORNER GROOVES / NECK GROOVES.....	325
J.	GROOVING CYCLES.....	326
K.	SPECIAL GROOVES	328
L.	GROOVES AND SUBPROGRAMS	328
29.	PART-OFF.....	331
A.	PART-OFF PROCEDURE.....	331
30.	SINGLE POINT THREADING	337
A.	THREADING ON CNC LATHES.....	337
B.	TERMINOLOGY OF THREADING	339
C.	THREADING PROCESS.....	340
D.	THREADING FEED AND SPINDLE SPEED	351
E.	TOOL REFERENCE POINT	356
F.	BLOCK-BY-BLOCK THREADING	357
G.	BASIC THREADING CYCLE - G92	359
H.	MULTIPLE REPETITIVE CYCLE - G76.....	361
I.	THREAD INFEEED METHODS	366
J.	COMPOUND INFEEED CALCULATIONS	369
K.	THREAD RETRACT MOTION	373
L.	HAND OF THREAD.....	376

M.	THREADING TO A SHOULDER.....	378
N.	OTHER THREAD FORMS	382
O.	TAPERED THREAD	383
P.	MULTISTART THREAD	389
Q.	THREAD RECUTTING.....	396
31.	SUBPROGRAMS.....	397
A.	MAIN PROGRAM AND SUBPROGRAMS	397
B.	SUBPROGRAM FUNCTIONS.....	399
C.	SUBPROGRAM NUMBERING	405
D.	SUBPROGRAM DEVELOPMENT.....	409
E.	MULTI LEVEL NESTING	413
F.	CONTOURING WITH A SUBPROGRAM	416
G.	TOOL CHANGE SUBPROGRAM.....	418
H.	100 000 000 HOLE GRID	419
32.	DATUM SHIFT	423
A.	DATUM SHIFT WITH G92 OR G50	423
B.	LOCAL COORDINATE SYSTEM.....	426
C.	MACHINE COORDINATE SYSTEM	429
D.	DATA SETTING.....	431
E.	WORK OFFSETS.....	433
F.	TOOL LENGTH OFFSETS	434
G.	CUTTER RADIUS OFFSETS	435
H.	LATHE OFFSETS.....	435
I.	MDI DATA SETTING	436
J.	PROGRAMMABLE PARAMETER ENTRY	437
33.	CNC LATHE ACCESSORIES.....	444
A.	CHUCK CONTROL.....	444
B.	TAILSTOCK AND QUILL.....	446
C.	BI-DIRECTIONAL TURRET INDEXING	448
D.	BARFEEDER ATTACHMENT.....	451
E.	ADDITIONAL OPTIONS.....	453
F.	PROGRAMMING EXAMPLE	455
34.	WRITING A CNC PROGRAM	458
A.	PROGRAM WRITING	458
B.	PROGRAM OUTPUT FORMATTING.....	460
C.	LONG PROGRAMS	466
35.	PROGRAM DOCUMENTS	470
A.	DATA FILES.....	470
B.	PROGRAM DOCUMENTATION.....	471
C.	SETUP AND TOOLING SHEETS	473
D.	DOCUMENTATION FILE FOLDER.....	477
36.	PROGRAM VERIFICATION.....	480
A.	DETECTION OF ERRORS	480
B.	GRAPHIC VERIFICATION	481
C.	AVOIDING ERRORS.....	482

D.	COMMON PROGRAMMING ERRORS	484
37.	CNC MACHINING	487
A.	MACHINING A NEW PART	487
B.	RUNNING THE FIRST PART	490
C.	PROGRAM CHANGES	492
D.	ALTERNATE MACHINE SELECTION	494
E.	MACHINE WARM UP PROGRAM	494
F.	CNC MACHINING AND SAFETY	496
G.	SHUTTING DOWN A CNC MACHINE	497
H.	EQUIPMENT MAINTENANCE	499
38.	INTERFACING TO DEVICES	500
A.	RS-232C INTERFACE	500
B.	TERMINOLOGY OF COMMUNICATIONS	501
C.	DATA SETTING	502
D.	CONNECTING CABLES	503
39.	MATH IN CNC PROGRAMMING	505
E.	BASIC ELEMENTS	505
F.	GEOMETRY	506
G.	POLYGONS	509
H.	TAPERS	511
I.	CALCULATIONS OF TRIANGLES	515
J.	ADVANCED CALCULATIONS	525
K.	CONCLUSION	525
40.	CNC AND CAD/CAM	526
A.	CAM SOFTWARE	526
B.	DESKTOP COMPUTER PROGRAMMING	527
C.	TOOLPATH GEOMETRY DEVELOPMENT	527
D.	TOOL PATH GENERATION	528
E.	COMPLETE ENVIRONMENT	528
F.	POST PROCESSORS	534
G.	IMPORTANT FEATURES	534
H.	SUPPORT AND MANAGEMENT	535

1. NUMERICAL CONTROL

Numerical Control technology as known today, emerged in the mid-20th century. It can be traced to the year of 1952, the U.S. Air Force, Massachusetts Institute of Technology in Cambridge, MA, USA, and the name of John Parsons (1913-2007), who is closely associated with the invention of numerical control. It was not applied in production manufacturing until the early 1960's. The real boom came in the form of CNC, around the year of 1972, and a decade later with the introduction of affordable microcomputers.

In manufacturing field, and particularly in metal working, Numerical Control technology has caused something of a revolution. Even in the days before computers became standard fixtures in every company and many homes, machine tools equipped with Numerical Control system found their special place in many machine shops. The relatively recent evolution of microelectronics and the never ceasing computer development, including its impact on Numerical Control, has brought enormously significant changes to manufacturing sector.

A. DEFINITION OF NUMERICAL CONTROL

The majority of all the known definitions can be summed up into a relatively simple statement:

“Numerical Control can be defined as an operation of machine tools by means of specifically coded instructions to the machine control system”

The '*specifically coded instructions*' are combinations of the letters of alphabet, digits, and selected symbols, for example, a decimal point, the percent sign, or the parenthesis symbols. All instructions are written in a logical order and in predetermined form. The collection of *all* instructions necessary to machine a single part or operation is called an *NC Program*, *CNC Program*, or a *Part Program*. Such a program can be stored for future use and used repeatedly to achieve identical machining results at any time.

NC and CNC Technology: *NC* stands for the older and original *Numerical Control* technology, whereby the abbreviation *CNC* stands for the newer *Computerized Numerical Control* technology - a modern successor to its older relative. However, in everyday practice, *CNC* is the preferred abbreviation. To clarify the proper usage of each term, look at the major differences between NC and CNC systems.

Both systems perform the same tasks, namely manipulation of data for the sole purpose of machining a part. In both cases, the control system internal design contains all logical instructions that process the input data. At this point the similarity ends.

The NC system (as opposed to the CNC system) uses a fixed logical function, those that are built-in and permanently wired within the control unit. These functions cannot be changed by the part programmer or the machine operator. Because of the fixed wiring of control logic, NC control system is synonymous with the term '*hardwired*'. The system can interpret a part program, but it does not allow any changes to the program at the control (using the control features). All required program changes must be made away from the control, typically in an office environment. Also, NC system typically requires the compulsory use of punched tapes for input of the program information.

The modern CNC system (but not the old NC system) uses an internal microprocessor (*i.e.*, a computer). This computer contains memory registers storing a variety of routines that can manipulate logical functions. That means the part programmer or machine operator can change any program at the control unit (at the machine), with instantaneous results. This flexibility is the greatest advantage of CNC systems and probably the key element that contributed to such a wide use of the technology in modern manufacturing. Typically, CNC programs and the logical functions are stored on special computer chips, as *software instructions*, rather than used by the hardware connections, such as wires, which control the logical functions. In contrast to the NC system, the CNC system is synonymous with the term '*softwired*'.

When describing a particular subject that relates to numerical control technology, it is customary to use either the term NC or CNC. Keep in mind that NC can also mean CNC in everyday talk, but CNC can never refer to the older

technology, described in this digital book under the abbreviation of NC. The letter 'C' stands for *Computerized*, and it is not applicable to the hardwired system. All control systems manufactured today are of the CNC design. Abbreviations such as C&C or C'n'C are not correct and reflect poorly on anybody that uses them.

B. CONVENTIONAL AND CNC MACHINING

What makes CNC machining methods superior to conventional methods? Are they superior at all? Where are the main benefits? While comparing CNC and conventional machining processes, common general approach to machining a typical part will emerge:

- Obtain and study the engineering drawing
- Select the most suitable machining method
- Decide on the setup method (work holding)
- Select cutting tools and holders
- Establish spindle speeds and cutting feed rates
- Machine the part

This general approach is the same for both types of machining. One major difference is *how* various data are input. A feed rate of 10 inches per minute (10 in/min) is the same in manual or CNC applications, but the method of applying it is not. The same can be said about a coolant - it can be activated by physically turning a knob, pushing a switch, *or* programming a special code. All these actions will result in coolant rushing out of a nozzle. In both kinds of machining, a certain amount of knowledge by the user is required. After all, metal working, and metal cutting specifically, is mainly a skill, but it is also, to a great degree, an art and a profession of large number of people. So is the application of *Computerized Numerical Control*. Like any skill, or art, or profession, mastering it to the last detail is necessary to be successful. It takes a lot more than just technical knowledge to be a CNC machinist, operator, or CNC programmer. Work experience, intuition, and what is sometimes called a '*gut-feel*', are much needed supplements to any skill.

In conventional machining, the operator sets up the machine and moves each cutting tool, using one or both hands, to produce the required part. Design of a manual machine tool offers many features that help the process of machining a part - levers, handles, gears, and dials, to name just a few. The same body motions are repeated by the operator for every part machined. However, the word '*same*' in this context really means '*similar*' rather than '*identical*'. Humans are not capable to always repeat every process the same - that is the job of machines. People cannot work at the same performance level all the time, without a rest. All of us have some good and some bad moments. Such moments, when applied to machining a part, are difficult to predict. There will always be some differences and inconsistencies within each batch of parts. Parts will not always be *the same*. Maintaining dimensional tolerances and surface finish quality are the most typical problems encountered in conventional machining. Individual machinists may have their own 'time proven' methods, different from those of their fellow colleagues. Combination of these and other factors create a large field of inconsistency.

Machining under numerical control does away with most inconsistencies. It does not require the same physical involvement as manual machining. Numerically controlled machining does not need any levers or dials or handles, at least not in the same sense as conventional machining. Once the part program has been proven, it can be used any number of times over, always returning consistent results. That does not mean there are no limiting factors. Cutting tools do wear out, material blank in one batch is not identical to the material blank in another batch, setups may vary, etc. These factors should be considered and compensated for, whenever necessary.

Emergence of numerical control technology does not mean an instant - or even a long term - demise of all manual machines. There are times when a traditional machining method is preferable to a computerized method. For example, a simple one-time job may be done more efficiently on a manual machine than on a CNC machine. Certain types of machining jobs will benefit from manual, semiautomatic or automatic machining, rather than machining under numerical control. CNC machine tools are not meant to replace every manual machine, only to supplement them.

In many instances, the decision whether certain machining will be done on a CNC machine or not is based on the number of required parts and nothing else. Although the volume of parts machined as a batch is always an important criterion, it should never be the only factor. Consideration should also be given to the part complexity, its tolerances, the required quality of surface finish, etc. Often, a single complex part will benefit from CNC machining, while fifty relatively simple parts will not.

Keep in mind that numerical control has never machined a single part by itself. Numerical control is only a *process* or a *method* that enables a machine tool to be used in a productive, accurate and consistent way.

C. NUMERICAL CONTROL ADVANTAGES

What are the main advantages of numerical control?

It is important to establish which areas of machining will benefit from it and which are better done the conventional way. It is absurd to think that a two horsepower CNC mill will win over jobs that are currently done on a twenty times more powerful manual mill. Equally unreasonable are expectations of super improvements in cutting speeds and feed rates over a conventional machine. If the machining and tooling conditions are the same, the total cutting time will always be very close in both cases.

A list of some major areas where CNC users can and should expect improvement includes:

Setup time reduction

Lead time reduction

Accuracy and repeatability

Contouring of complex shapes

Simplified tooling and work holding

Consistent cutting time

General productivity increase

Each area offers only a *potential* improvement. Individual CNC users will experience different levels of *actual* improvement, depending on the product manufactured, CNC machine used, setup methods applied, complexity of fixturing, quality of cutting tools, management philosophy and engineering design, experience level of the workforce, individual attitudes, and many others.

Setup Time Reduction

In many cases, actual setup times for CNC machines can be reduced, sometimes quite dramatically. It is important to realize that setup is a manual operation, greatly dependent on the performance of CNC operators, the type of fixturing and general machine shop practices. Setup time is unproductive, but necessary - it is part of the overall costs of doing business. To keep setup time to minimum should be the primary consideration of any machine shop supervisor, programmer, and operator.

Because of the design of CNC machines, real setup time should not be a major problem. Modular fixturing, standardized tooling, fixed locators, automatic tool changing, pallets, and other advanced features, make the setup time more efficient than a comparable setup of conventional machines. With good knowledge of modern manufacturing, productivity can be increased quite

significantly.

The number of parts machined in a single setup is also important, in order to assess the actual cost of setup time. If a great number of parts is machined in one setup, the setup cost per part can be rather insignificant. A very similar reduction can be achieved by grouping several different operations into a single setup. Even if the setup time is longer, it may be justified when compared to the time required to setup several conventional machines and operations.

Lead Time Reduction

Once a part program is written and proven correct, it is ready to be used again in the future, even at a short notice. Although the first run lead time is usually longer, it is virtually nil for all subsequent runs. Even if an engineering change of the part design requires program modification, it can be done usually quickly, reducing the lead time.

Long lead time, required to design and manufacture several special fixtures for conventional machines, can often be reduced by using simplified fixturing.

Accuracy and Repeatability

The high degree of accuracy and repeatability of modern CNC machines has been the single major benefit to many users. Whether part program is stored on a disk or in the computer memory, or even on a tape (the original method, now obsolete), it always remains the same. Any program can be changed at will, but once proven, no changes are usually required any more. A given program can be reused as many times as needed, without losing a single bit of data it contains. True, program has to allow for such changeable factors as tool wear and operating temperatures, it has to be stored safely, but generally very little interference from the CNC programmer or operator will be required. The accuracy of modern CNC machines and their repeatability allows high quality parts to be produced consistently, time after time.

Contouring of Complex Shapes

CNC lathes and machining centers can contour a large variety of different shapes. Many CNC users acquired their machines only to be able to handle complex parts. Good examples are CNC applications in the aircraft and automotive industries. Any use of some kind of computerized programming is virtually mandatory for any three-dimensional tool path generation.

Complex shapes, such as molds, manifolds, dies, etc., can be manufactured without the additional expense of making a model for tracing. Mirrored parts can be achieved literally at the switch of a button. Storage of part programs is a lot simpler than storage of paper patterns, templates, wooden models, and other pattern making tools.

Simplified Tooling and Work Holding

Non-standard and 'homemade' tooling that clutters the benches and drawers around a conventional machine can be eliminated by using standard tooling, specially designed for numerical control applications. Multi-step tools such as pilot drills, step drills, combination tools, counter borers and others, are replaced with several individual standard tools. These tools are often cheaper and easier to replace than special and non-standard tools. Cost-cutting measures have forced many tool suppliers to keep a low or even a nonexistent inventory, while increasing delivery time to the customer. Standard, off-the-shelf tooling can usually be obtained faster than non-standard tooling.

Fixturing and work holding for CNC machines have only one major purpose - to hold the part rigidly and in the same position for all parts within a batch. Fixtures designed for CNC work do not normally require special jigs, pilot holes and other holes locating aids.

Cutting Time and Productivity Increase

Cutting time on a CNC machine is commonly known as the *cycle time* - and is always consistent. Unlike a conventional machining, where the operator's skill, experience and personal fatigue are subject to changes, CNC machining is under the control of a computer. Only a small amount of manual work is restricted to the setup and part loading and unloading. For large batch

runs, the high cost of unproductive time is spread among many parts, making it less significant. The main benefit of a consistent cutting time is for repetitive jobs, where production scheduling and work allocation to individual machine tools can be done very efficiently and accurately.

One of the main reasons companies often purchase CNC machines is strictly economic - it is a serious investment with great potential. Also, having a competitive edge is always on the mind of every plant manager. Numerical control technology offers excellent means to achieve significant improvements in manufacturing productivity and increasing the overall quality of manufactured parts. Like any means to an end, it has to be used wisely and knowledgeably. When more and more companies use CNC technology, just having a CNC machine does not offer the extra edge anymore. Companies that grow and get forward are those where the use of technology is managed efficiently, with the goal to be competitive in the global economy.

To reach the goal of major increase in productivity, it is essential that users understand the fundamental principles on which CNC technology is based. These principles take many forms, for example, understanding the electronic circuitry, complex ladder diagrams, computer logic, metrology, machine design, machining principles and practices, and many others. Each discipline must be studied and mastered by all persons in charge. In this digital book, the main emphasis is on topics relating directly to CNC programming and understanding the most common CNC machines - *Machining Centers* and *Lathes* (sometimes called *Turning Centers*). Part quality consideration should be very important to every programmer and machine operator and this goal is also reflected in the digital book approach as well as in numerous examples.

D. TYPES OF CNC MACHINE TOOLS

Different types of CNC machines cover rather large variety. Number of installations is rapidly increasing, and the technology development advances at a rapid pace. It is impossible to identify all possible applications, they would make a long list. Here is a brief list of some of the groups CNC machines can be part of:

- Mills and Machining centers
- Lathes and Turning centers
- Drilling machines
- Boring mills and Profilers
- EDM wire machines
- Punch presses and Shears
- Flame cutting machines
- Routers
- Water jet and Laser profilers
- Cylindrical grinders
- Welding machines
- Benders, Winding
- Spinning machines, etc.

Without a doubt, CNC machining centers and lathes dominate the number of installations in industry. These two groups share the market just about equally. Some industries may have a higher need for a particular type of machines, depending on their needs. One must remember there are many models of lathes available and equally many different models of machining centers. However, the programming process for a vertical machine is similar to the one for a horizontal machine or even a simple CNC mill, for example. Even between different machine groups, there is a great number of general applications, while the programming process is generally unchanged. For example, a contour milled with an end mill has a lot in common with a contour cut with a wire on an EDM machine.

E. LATHES AND TURNING CENTERS

A CNC lathe in its basic form is a machine tool with two axes, vertical X axis and horizontal Z axis. The main feature of a lathe that distinguishes it from a mill is that the part is rotating about the machine center line. In addition, the cutting tool is normally stationary, mounted in a sliding turret. Cutting tool follows the contour of the programmed tool path. Many modern CNC lathes are much more than just turning centers - with a simple milling attachment, the so-called *live tooling*, the milling cutter has its own motor and rotates while the spindle is stationary. More complex designs incorporate off-center milling, double spindles, double turrets, part transfer, and

many other efficiency improving features. These machines are generally called the *mill-turn* centers or sometimes the *turn-mill* centers.

Modern lathe design can be horizontal or vertical. Horizontal type is far more common than vertical type, but both designs have their purpose in manufacturing. Several different designs exist for either group. For example, a typical CNC lathe of the horizontal group can be designed with a flat bed or a slant bed, as a bar type, chucker type or a universal type. Added to these combinations are many accessories that make a CNC lathe an extremely flexible machine tool. Typically, accessories such as tailstock, steady rests or follow-up rests, part catchers, pullout-fingers and a third axis milling attachment are popular components of CNC lathes. A CNC lathe can be very versatile - so versatile in fact, that it is often called a *CNC Turning Center*. All text and program examples in this digital book use the more traditional term *CNC lathe*, yet still recognizing all its modern functions.

F. PERSONNEL FOR CNC

Computers and machine tools have no intelligence. They cannot think, they cannot evaluate a situation in a rational way. Only people with certain skills and knowledge can do that. In the field of numerical control, the skills are usually in the hands of two key people - one doing the *programming*, the other doing the actual *setup* and *machining*. Their respective numbers and duties typically depend on company preferences, its size, as well as the product manufactured there. However, each position is quite distinct, although many companies combine the two functions into a one, often called a *CNC Programmer/Operator*.

CNC Programmer

CNC programmer is usually a person who has the most responsibility in the CNC machine shop. This person is often responsible for the success of numerical control technology in whole the plant. Equally, this person is also held responsible for problems related to CNC and related operations. Although duties may vary, the programmer is also responsible for a variety of tasks relating to the effective usage of one or more CNC machines. In fact, this person is often accountable for the production and quality parts from all CNC operations.

Many CNC programmers are experienced machinists who have had a practical, hands-on experience as machine tool operators. They know how to read technical drawings and they can comprehend engineering intent behind the design. This practical experience is the main foundation for the ability to '*machine*' a part in an off-machine environment. A good CNC programmer must be able to visualize all tool motions and recognize all restricting factors that may be involved. The programmer must be able to collect, analyze, process and logically integrate all collected data into a single, cohesive and safe part program. In simple terms, the CNC programmer must be able to decide upon the best manufacturing methodology in all respects.

In addition to machining skills, the CNC programmer has to have a good understanding of mathematical principles, mainly application of equations, solution of arcs and angles. Equally important is the knowledge of trigonometry. Even with computerized programming, the knowledge of manual programming methods is absolutely essential to deep and thorough understanding of computer output and to assure control over such output.

The last important quality of a truly professional CNC programmer is his or her ability to listen to other people the engineers, CNC operators, managers. Good listening skills are the first prerequisites to become flexible. Any professional CNC programmer must be flexible to offer high quality in programming.

CNC Machine Operator

The CNC machine tool operator is a complementary position to that of CNC programmer. The programmer and the operator may exist in a single person, as is common in many smaller shops. Although most duties performed by a conventional machine operator has been transferred to CNC

programmer, CNC operator also has many unique responsibilities. In typical cases, the operator is responsible for tool and machine setup, for changing of completed parts, often even for some in-process inspection. Many companies expect quality control at the machine and the operator of any machine tool, manual or computerized, is also responsible for the quality of work done on that machine. One of the most important responsibilities of CNC machine operator is to report findings about each program to the programmer. Even with the best knowledge, skills, attitudes, and good intentions, the 'final' program can always be improved. CNC operator, being the one who is the closest to actual machining, knows precisely what extent such improvements can be.

G. SAFETY RELATED TO CNC WORK

On the walls of many machine shops may hang a safety poster with a simple, yet very powerful message: The first rule of safety is to follow all safety rules.

The heading of this section does not indicate whether safety is oriented at the programming or the machining level. There is no reason for it - safety is totally *independent*. It stands on its own and it governs behavior and activities of everybody in machine shop and outside of it. At first sight, it may appear that safety is something related to machining and machine operations, perhaps to the machine setup as well. That is true but hardly presents a complete picture.

Safety is the most important element in programming, setup, machining, tooling, fixturing, inspection, shipping, and *you-name-it* operation within a typical machine shop daily work. Safety should never be compromised and cannot be overemphasized. Companies talk about safety, conduct safety meetings, display posters, make speeches, call experts. This mass of information and instructions is presented to all of us for some very good reasons. Quite a few are based on past tragic occurrences - many laws, rules and regulations have been written because of inquests and inquiries into serious accidents.

At first sight, it may seem that in CNC work, safety is a secondary issue, not as important as in manual machining. There is a lot of automation in CNC, a part program that is used repeatedly, tooling that has been used in the past, a simple setup, etc. All this can lead to complacency and false assumption that safety is taken care of. This is a wrong view that can have serious consequences.

Safety is quite a large subject but a few points that relate to CNC environment are very important. Every machinist should know the hazards of mechanical and electrical devices. The first step towards a safe workplace is with a clean work area, where no chips, oil spills and other debris are allowed to accumulate on the floor. Taking care of personal safety is equally important. Loose clothing, jewelry, ties, scarfs, unprotected long hair, improper use of gloves and similar infractions, is dangerous in any machining environment. Protection of one's eyes, ears, hands and feet are strongly recommended.

While a machine is operating, protective devices should be in place and no moving parts should be exposed. Special care should be taken around rotating spindles and automatic tool changers. Other devices that could pose a hazard are pallet changers, chip conveyors, high voltage areas, hoists, etc. Disconnecting any interlocks or other safety features is dangerous - and illegal, without appropriate skills and authorization.

Modern technology has brought machines that may have nine or more axes, tight work areas, special tool indexing, part transfers, etc. While these features dramatically increase company productivity, they also require additional safety training - and practicing all safety rules.

In CNC programming (manual or computer based), observation of safety rules is equally important. A tool motion can be programmed in many ways. Speeds and feeds must be realistic, not just mathematically 'correct'. Depth of cut, width of cut, various tool characteristics, they all have a profound effect on overall safety in the shop.

All these ideas are just a very short summary and a reminder that safety should be always taken

seriously.

2. COORDINATE GEOMETRY

One of the first major steps towards basic understanding of CNC principles and geometrical concepts is thorough understanding of a subject known in mathematics as the *system of coordinates*. The most important of these principles are those that can be applied to CNC technology of today. In various publications on mathematics and geometry, these principles are often listed under the headings such as the *real number system* and the *rectangular coordinates*.

One main key to understanding rectangular coordinates is understanding of basic arithmetic, algebra, and geometry. The key knowledge in this area is knowledge of the *real number system*. Within the real number system, there are ten available numerals (digits), 0 to 9 (zero to nine), that can be used in any of the following groups:

Zero integer	0
Positive integers (<i>with or without sign</i>)	1, 2, +3, 10, 12943, +45
Negative integers (<i>minus sign required</i>)	-4, -381, -25, -77
Fractions	1/8, 3/16, 9/32, 35/64
Decimal fractions	0.185, 0.2, 0.546875, 3.5, 15.0

All groups are used daily. These groups represent the mainstream of just about all applications of numbers in modern life. In CNC programming, the primary goal is to use numbers to ‘translate’ engineering drawing based on its dimensions - into a specific cutter path.

Computerized Numerical Control means *control by the numbers using a computer*. All drawing information must be translated into CNC program, using primarily numbers. Numbers are also used to describe commands, functions, comments, and so on. The mathematical concept of a real number system can be expressed graphically on a horizontal or a vertical line, called the *number scale*, where all divisions have the same length - *Figure 1*

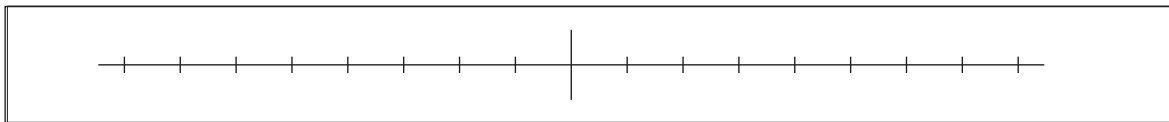


Figure 1: Graphical representation of the Number Scale

Length of each division on the scale represents the unit of measurement in a convenient and generally accepted scale. It may come as a surprise that this concept is used every day. For example, a simple ruler used in schools is based on the number scale concept, regardless of measuring units. Weight scales using tons, pounds, kilograms, grams, and similar units of mass are other examples. A simple household thermometer uses the same principle. Other similar examples are available as well.

A. RECTANGULAR COORDINATE SYSTEM

Rectangular coordinate system is a concept used to define a planar 2D point (two dimensions), using the XY coordinates, or a spatial 3D point (three dimensions), using the XYZ coordinates. This system was first defined in the 17th century by a French philosopher and mathematician *Rene Descartes (1596-1650)*. His name is used as an alternative name of the rectangular coordinate system, it is called the *Cartesian Coordinate System* - see *Figure 2*.

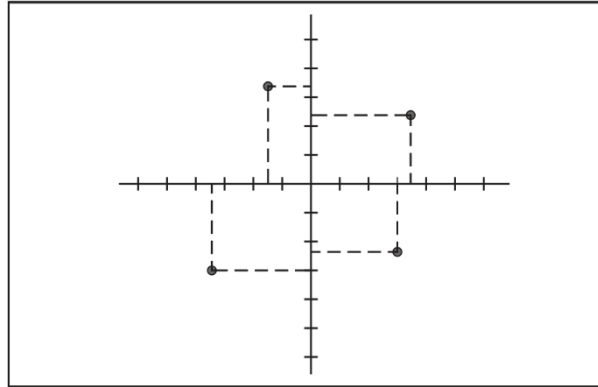


Figure 2: Rectangular coordinate system = Cartesian coordinate system

The concepts used in design, drafting *and* in numerical control are over 400 years old. A given point can be mathematically defined on a plane (two coordinate values) or in space (three coordinate values). The definition of one point is relative to another point as a distance parallel with one of three axes that are perpendicular to each other. In a plane, only two axes are required, in space, all three axes must be specified. In programming, point represents an exact location. If such a location is on a plane, the point is defined as a 2D point, along two axes. If the location is in space, the point is defined as a 3D point, along three axes.

When two number scales that intersect at right angles are used, mathematical basis for a *rectangular coordinate system* is created. Several terms emerge from this representation, and all have an important role in CNC programming. Their understanding is very important for further progress.

Axes and Planes

Each major line of the number scale is called an *axis*. It could have either vertical or horizontal orientation. This very old principle, when applied to CNC programming, means that at least two axes - *two number scales* - will be used. This is the mathematical definition of an axis:

An axis is a straight line passing through the center of a plane or a solid figure, around which the parts are symmetrically arranged.

This definition can be enhanced by a statement that an axis can also be a *line of reference*. In CNC programming, an axis is used as a reference all the time. The definition also contains the word '*plane*'. A *plane* is a term used in 2D applications, while a *solid object* is used in 3D applications. Mathematical definition of a plane is: A plane is a surface in which a straight line joining any two of its points will lie wholly on the surface.

From the *top viewpoint* of the observer, looking straight down on the illustration *Figure 3*, a *viewing direction* is established. This is often called *viewing a plane*.

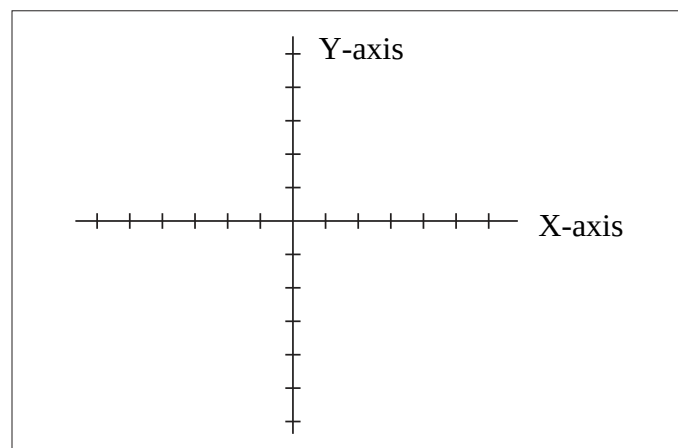


Figure 3: Axis designation - viewing a plane

Mathematical designation is fully implemented in CNC

A plane is a 2D entity - the letter X identifies its horizontal axis; the letter Y identifies its vertical axis. Such plane is called the *XY plane*. Defined mathematically, the horizontal axis is *always* listed as the *first* letter of the pair. In drafting and CNC programming, this plane is also known as the *Top View* or a *Plan View*. Other planes are also used in CNC, but not to the same extent as in CAD/CAM work.

Point of Origin

Another term that emerges from the rectangular coordinate system is called the *point of origin*, or just *origin*. It is the exact point where the two perpendicular axes intersect. This point has a zero-coordinate value in each axis, specified as planar X0Y0 and spatial X0Y0Z0 - *Figure 4*.

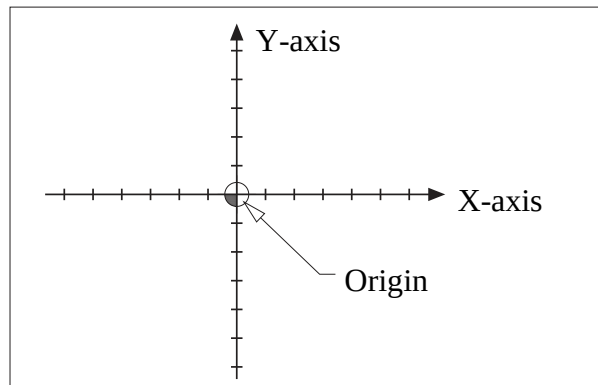


Figure 4: Point of origin - intersection of axes

This intersection called *origin* has a rather special meaning in CNC programming. This origin point acquires a new name, one typically called the *program reference point*. Other terms are also used: *program zero*, *part reference point*, *workpiece zero*, *part zero*, and probably a few others - all with the same meaning and same purpose.

Quadrants

Viewing the two intersecting axes and the new plane, four distinct areas can be clearly identified. Each area is bounded by two axes. These areas are called *quadrants*.

Mathematically defined, a quadrant is any one of the four parts of the plane formed by the system of rectangular coordinates. It suggests *four* uniquely defined areas or *quadrants*. Looking down in the top view at the two intersecting axes, the following definitions apply to quadrants. They are mathematically correct and are used in all CNC/CAD/CAM applications:

Quadrant I	UPPER RIGHT
Quadrant II	UPPER LEFT
Quadrant III	LOWER LEFT
Quadrant IV	LOWER RIGHT

Quadrants are always defined in the *counterclockwise* direction, from the horizontal X-axis and the naming convention uses *Roman* - not *Arabic* - numbers normally used.

COORDINATE GEOMETRY

Numbering of quadrants always starts at the *positive* side of the *horizontal* axis. *Figure 5* illustrates the definitions.

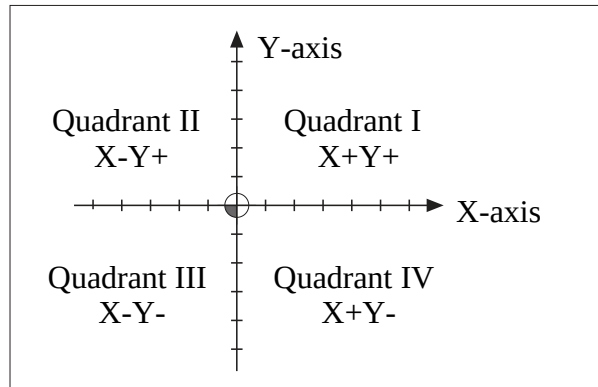


Figure 5: Quadrants in the XY plane and their identification

Any point coordinate value can be positive, negative or zero. All point coordinates are determined solely by their *location* in a particular quadrant and individual distances along an axis, again, relative to origin - Figure 6.

Point location	Coordinate	
	X-axis	Y-axis
Quadrant I	+	+
Quadrant II	-	+
Quadrant III	-	-
Quadrant IV	+	-

Figure 6: Algebraic signs for a point location in plane quadrants

IMPORTANT:

If the defined point lies exactly on the X-axis, it has the Y value equal to zero (Y0)

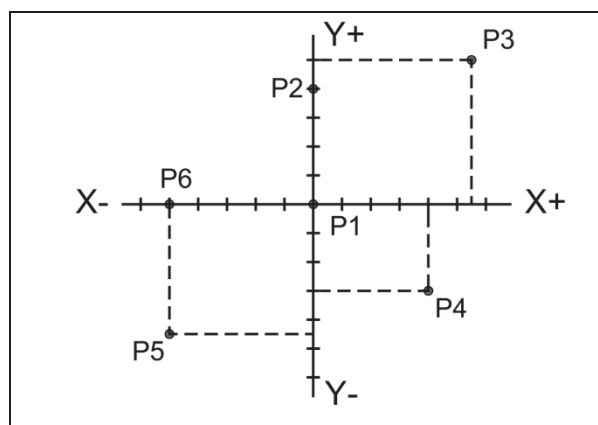
If the point lies exactly on the Y-axis, it has the X value equal to zero (X0)

If the point lies exactly on both X and Y axes, both X and Y values are zero (X0 Y0).

X0Y0Z0 is the point of origin. In part programming, positive values are written without the plus sign - Figure 7.

Right Hand Coordinate System

In all illustrations of *number scale*, *quadrants* and *axes*, the *origin* divides each axis into two portions. The zero point - *the point of origin* - separates the *positive* section of an axis from the *negative* section. In the right-hand coordinate system, the *positive* axis starts at origin and is directed towards the *right* for X-axis, *upwards* for Y-axis and *towards the perpendicular viewpoint* for Z-axis. Opposite directions are always negative.



P1 = X0.0 Y0.0	P4 = X4.0 Y-3.0
P2 = X0.0 Y4.0	P5 = X-5.0 Y-4.5
P3 = X5.5 Y5.0	P6 = X-5.0 Y0.0

Figure 7: Coordinate definition of points within rectangular coordinate system

Point P1 = Origin = X0Y0

If these directions were superimposed over a human right hand, they would correspond to the direction from the root of thumb or finger towards its tip. Thumb would point in the X+ direction, index finger in the Y+ direction and middle finger in the Z+ direction.

CNC machines are normally programmed using the so-called *absolute* coordinate method, which is based on the point of origin being X0Y0Z0. This absolute programming method follows very strictly the rules of rectangular coordinate geometry and all concepts covered in this chapter.

B. MACHINE GEOMETRY

Machine geometry defines the relationship of distances and dimensions between *fixed point of the machine* and *selectable point of the part*. Typical geometry of CNC machines uses the right-hand coordinate system. Positive and negative axis direction is determined by an established viewing convention. The general rule for Z-axis is that it is always the axis along which a simple hole can be machined with a single point tool, such as a drill, reamer, wire, laser beam, etc. *Figure 4-8* on the next page illustrates standard orientation of planes for XYZ type machine tools.

Axis Orientation:

A typical machining center has three controlled axes, defined as X-axis, Y-axis, and Z-axis. X-axis is parallel to the longest dimension of machine table; Y-axis is parallel to the shortest dimension of the table and Z-axis is the spindle movement. On a vertical CNC machining center, X-axis is the table longitudinal direction, Y-axis is the saddle cross direction, and Z-axis is the *spindle* direction.

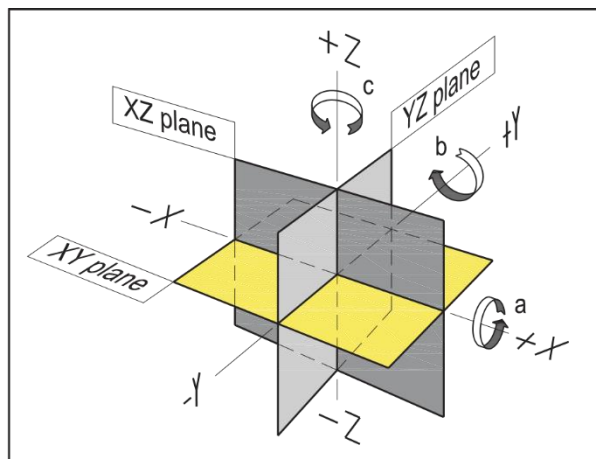


Figure 8: Standard orientation of planes and CNC machine tool axes

Axis Orientation - Turning

Standard CNC lathes have two axes, X and Z. More axes are available, but they are not important at this point. Special additional axes, such as C-axis and Y-axis, are designed for milling operations (live tooling) and require unique version of a standard CNC lathe.

What is much more common for CNC lathes in industry, is the double orientation of XZ axes. CNC lathes are separated as *front* and *rear* lathes. An example of a front lathe is like the conventional engine lathe. All slant bed lathe types are of the rear kind. Identification of axes in industry have not always followed mathematical principles.

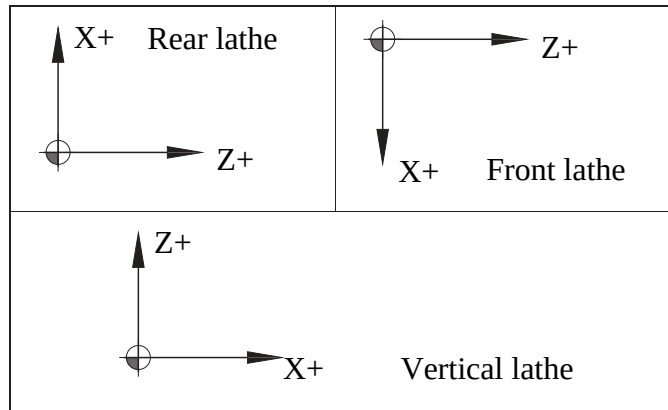


Figure 9: Typical machine axis orientation for various CNC lathes

Another lathe variety, a vertical CNC lathe, is basically a horizontal lathe rotated 90°. Typical axes for horizontal and vertical machine axes, as applied to turning, are illustrated in *Figure 9*.

Additional Axes

A CNC machine of any type can be designed with one or more additional axes, normally designated as the secondary - or *parallel* - axes using the U, V and W letters. These axes are normally parallel to the primary X, Y and Z axes respectively. For a rotary or an indexing application, additional axes are defined as A, B and C axes, as being rotated about the X, Y and Z axes, again in their respective order. Positive direction of a rotary (or an indexing) axis is the direction required to advance a right-handed screw in the positive X, Y or Z axis. Relationship between the primary and supplementary axes is shown in *Figure 10*.

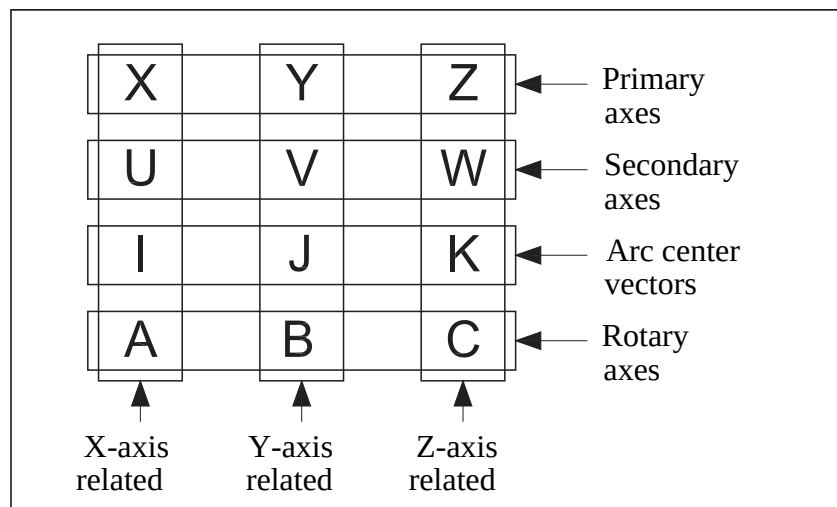


Figure 10: Relationship of primary and supplementary machine axes

Arc center modifiers (sometimes called arc center vectors) are not true axes, yet they are also related to primary axes XYZ. This subject will be described in the section on *Circular Interpolation*.

3. CONTROL SYSTEM

A machine unit equipped with a computerized numerical control system is commonly known as a CNC machine. In an analogy of the machine tool being the *body* of a CNC machine system, the control unit is its *brain*, its nerve center. There are no levers, no knobs, and no handles on a CNC machine the way they function on conventional lathes. All machine speeds, feeds, axes motions and hundreds of other tasks are programmed by a CNC programmer and controlled by a computer that is major part of the CNC unit. To make a program for a CNC machine tool means to make a program for the control system. True, the machine tool is a major consideration as well, but it is the *control unit* that determines the program *format*, its *structure* and its *syntax*.

To fully understand CNC programming process, it is important to understand not only the intricacies of how to machine a part, what tools to select, what speeds and feeds to use, how to setup the job and many other features. It is equally important to know how the computer, the CNC unit works without the need to be an expert in electronics or a computer scientist. *figure 11* shows an actual Fanuc control panel.

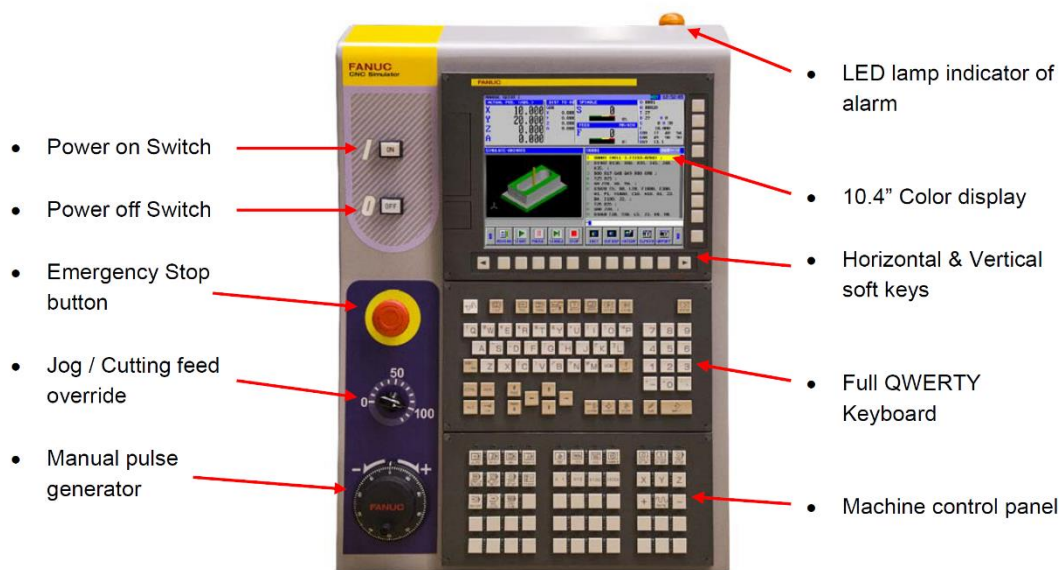


Figure 11: A example of a Fanuc control panel - actual layout and features will vary on different models

Machine manufacturers add their own operation panel, with all switches and button needed to operate their CNC machine and all its features. A typical operation panel is illustrated in Figure 12. Another item required for the system, *the handle*, will be described as well.

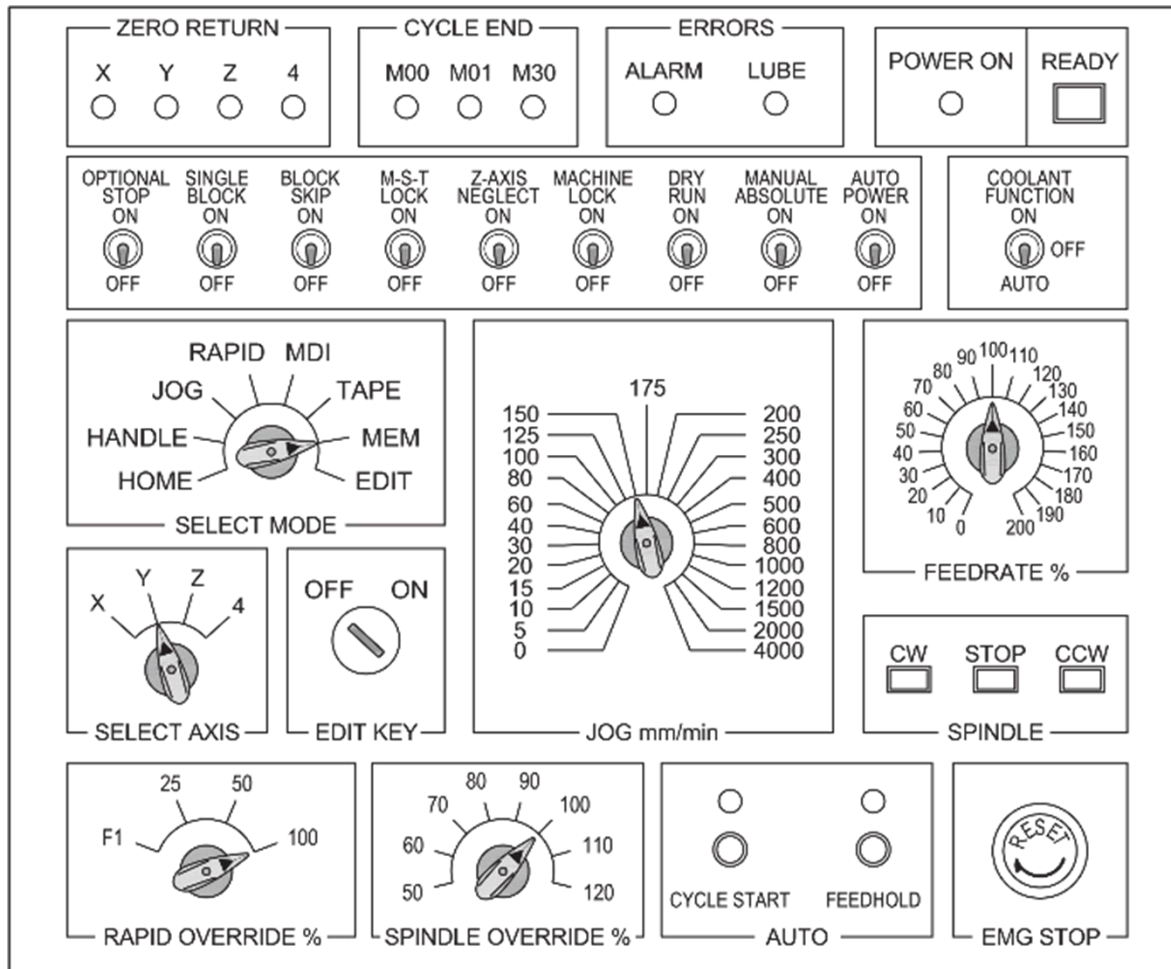


Figure 12: A operation panel of a CNC machining - actual layout and features will vary on different models

Feature	Description
ON / OFF switch	Power and control switch for the main power and control unit
Cycle Start	Starts program execution or MDI command
Emergency Stop	Stops all machine activity and turns off power to the control unit
Feed hold	Temporarily stops motion of all axes
Single Block	Allows program run one block at a time
Optional Stop	Temporarily stops program execution (M01 required in program)
Block Skip	Ignores blocks preceded with a forward slash (/) in the program
Dry Run	Enables program testing at fast feed rates (without a mounted part)
Spindle Override	Overrides programmed spindle speed, usually within 50-120% range
Feed rate Override	Overrides programmed feed rate, usually within 0-200% range
Chuck Clamp	Shows current status of the chuck clamping (Outside / Inside clamping)

Feature	Description
Table Clamp	Shows current status of table clamping
Coolant Switch	Coolant control ON / OFF / AUTO
Gear Selection	Shows current status of working gear range selection
Spindle Rotation	Indicates spindle rotation direction (clockwise or counterclockwise)
Spindle Orientation	Manual orientation of the spindle
Tool Change	Switch allowing a manual tool change
Reference Position	Switches and lights relating to setup of machine, from reference position
Handle (MPG)	Manual Pulse Generator (MPG), used for Axis Select and Handle Increment switches
Tailstock Switch	Tailstock and/or quill switch to manually position the tailstock
Indexing Table Switch	Manually indexes machine table during setup
MDI Mode	Manual Data Input mode

A. GENERAL DESCRIPTION

Even a brief look at any control unit reveals that there are two basic components - one is the *operation panel*, full of rotary switches, toggle switches and push buttons. The other component is the *display screen* with a keyboard or a keypad. A programmer who does not normally work on the CNC machine will seldom, if ever, have a reason to use either the operation panel or the display screen. They are available at the machine to the CNC machine operator and used for machine setup as well as to control the activities of the machine.

Should the CNC programmer be at least interested in the machine operation? Is it necessary for the programmer to *know* and *understand* all functions of the control system? There is only one answer to both questions - *definitely yes*.

The control unit

The CNC system - contains features that only work in conjunction *with* the program, it does not do anything useful on its own. Some features can be used *only* if the program itself supports them. All switches and buttons and keys are used by the machine operator, to exercise control over program execution and machining process.

Operation Panel

Depending on CNC machine type, the following table covers the most typical and common features found on modern operation panel. There are some small differences for operation of a machining center and a lathe, but both operation panels are similar. It is always a good idea to double check with the manufacturer specifications and recommendations. It is common that many machines used in the shop have some special features.

Even if some features may not be listed, virtually all of those in the table are somewhat related to CNC program. Many control systems have unique features of their own. These features must be known to the CNC operator. The program supplied to the machine should be *flexible*, not *rigid* - it should be 'user friendly'.

Screen Display and Keyboard

Feature	Description
AUTO-mode	Allows automatic operations
MEMORY mode	Allows program execution from memory of the CNC unit
TAPE / EXT or DNC mode	Allows program execution from an external device, such as a desktop computer (DNC) or a punched tape
EDIT mode	Allows changes to be made to a program stored in CNC memory
MANUAL Mode	Allows manual operations during setup
JOG Mode	Selects jog mode for setup
RAPID Mode	Selects rapid mode for setup
Memory Access	Key (switch) to allow program editing
Error Lights	Red light indicating an error

Screen display is the 'window' to control operation. Any active program can be viewed, including the control status, current tool position, various offsets, parameters, even a graphic representation of the toolpath. On all CNC units, individual monochrome or color screens can be selected to have the desired display at any time, using the input keys (keyboard pads and soft keys). Setting for international languages is also possible.

Keyboard pads and soft keys are used to *input* instructions to the control. Existing programs can be modified or deleted; new programs can be added. Using keyboard input, not only the machine axes motion can be controlled, but the spindle speed and feed rate as well. Changing internal parameters and evaluating various diagnostics are more specific means of control, often restricted to service people. Keyboard and screen are used to set program origin and to hook up to external devices, such as a connection with another computer. There are many other options, particularly for multi axis machines. Every keyboard allows the use of *letters*, *digits*, and *symbols* for data entry. Not every keyboard allows the use of *all* alphabet letters or *all* available symbols. Some control panel keys have a description of an *operation*, rather than a letter, digit or symbol, for example, *Read* and *Punch* Keys or the *Offset* key.

Handle

For setup purposes, each CNC machine has a rotary handle that can move one selected axis by as little as the least increment of the control system. The official Fanuc name for the handle is *Manual Pulse Generator*. Associated with the handle is *Axis Select* switch (often duplicated on the operation panel as well as on the handle) and the range of increment (that is the least increment X1, X10 and X100). The letter X in this case is the multiplier and stands for '*X times*'. One handle division will move the selected axis by X times the minimum increment of active units of measurement. Handles with digital display on a small screen are also available. In *Figure 13* and the following table are details of a typical traditional handle.

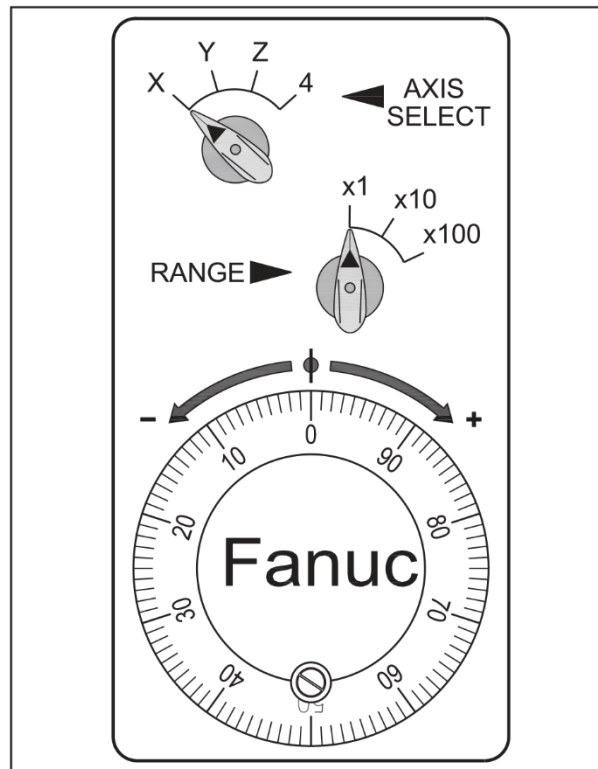


Figure 13: An example of a detached handle, called the Manual Pulse Generator (MPG), with a typical layout and features

Layout and features may vary on different machine models.

Handle multiplier	One handle division motion is ...	
	for metric units	for imperial units
X1	0.001 mm	0.0001 inch
X10	0.010 mm	0.0010 inch
X100	0.100 mm	0.0100 inch

B. SYSTEM FEATURES

CNC unit is nothing more than a sophisticated special purpose computer. The 'special purpose' in this case is a *computer capable of controlling the activities of a machine tool*, such as lathe or machining center. It means the computer has to be designed by a company that has expertise in this type of special purpose computers. Unlike many business types of computers, each CNC unit is made for a particular customer. The customer is typically a machine manufacturer, *not* the end user. The manufacturer specifies certain requirements that the control system has to meet, requirements that reflect uniqueness of the machines they build. Basic control does not change, but some customized

features may be added (or taken away) for a specific machine. Once a control system is sold to the machine manufacturer, more features are added to the whole system. They relate to the design and capabilities of the machine.

A good example is a CNC unit for two machines that are the same in all respects except one. One machine has a *manual* tool changer, the other has an *automatic* tool changer. To support the automatic tool changer, the CNC unit must have special features built-in, that are not required for

a machine without the tool changer. The more complex a CNC system is, the more expensive it is. Users that do not require all sophisticated features, do not pay a premium for features they do not need.

Parameter Settings

Information that establishes the built-in connection between the CNC control and the machine tool is stored as special data in internal registers, called *system parameters*. Some of the information in this digital book is quite specialized and listed for reference only. Programmers with limited experience do not need to know system parameters in a great depth. Original factory settings are sufficient for most machining jobs.

When the parameter screen is displayed, it shows the parameter number with some data in a row. Each row of numbers represents one *byte*, each digit in the byte is called a *bit*. The word *bit* is made from the words *Binary digit* and is the smallest unit of a parameter input. Numbering of bits starts with 0, read from *the right to the left*:

Number	#7	#6	#5	#4	#3	#2	#1	#0
xxxx	0	1	1	0	1	0	0	1

Fanuc control system parameters belong to one of three groups, specified within an allowed range: **Binary codes**, Units **inputs**, Setting **values**

These groups use different input values. The *binary input* can only have an input of a 0 or 1 for the *bit* data format, 0 to +127 for the byte type. *Units input* has a broader scope units can be in mm, inches, mm/min, in/min, degrees, milliseconds, etc. A *value* can also be specified within a given range, for example, a number within the range of 0-99, or 0-99999, or +127 to -127, etc.

Atypical example of a *binary input* is a selection between *two options*. For instance, a feature called *dry run* can be set only as *effective* or *ineffective*. To select a preference, an arbitrary bit number of a parameter has been set to 0 to make the dry run effective and to 1 to make it ineffective.

Units input, for example, is used to set the increment system - the dimensional units. Computers in general do not distinguish between imperial and metric units, just numbers. It is up to the user *and* the parameter setting, whether the control will recognize 0.001 mm or 0.0001 inches as the least increment. Another example is a parameter setting that stores the maximum feed rate for each axis, the maximum spindle speed, etc. Such values must never be set higher than the machine itself can support. An indexing axis with a minimum increment of 1, will *not* become a rotary axis with 0.001° increment, just because the parameter is set to a lower value, even if it is possible. **Such a setting is wrong and can cause serious damage!**

To better understand what CNC system parameters can do, here is an abbreviated listing of parameter classification for a typical Fanuc control system (many of them are meaningful to the service technicians only):

Parameters related to Setting

Parameters related to Axis Control Data

Parameters related to Chopping

Parameters related to the Coordinate System

Parameters related to Feed rate

Parameters related to Acceleration/Deceleration Control

Parameters related to Servo

Parameters related to DI/DO

Parameters related to MDI, EDIT, and CRT

Parameters related to Programs

Parameters related to Serial Spindle Output

Parameters related to Graphic Display Parameters related to I/O interface

Parameters related to Stroke Limit

Parameters related to Pitch Error Compensation

Parameters related to Inclination Compensation

Parameters related to Straightness Compensation

Parameters related to Spindle Control

Parameters related to Tool Offset

Parameters related to Canned Cycle

Parameters related to Scaling and Coordinate Rotation

Parameters related to Automatic Corner Override

Parameters related to Involute Interpolation

Parameters related to Uni-directional Positioning

Parameters related to Custom Macro (User Macro)

Parameters related to Program Restart

Parameters related to High-Speed Skip Signal Input

Parameters related to Automatic Tool Compensation

Parameters related to Tool Life Management

Parameters related to Turret Axis Control

Parameters related to High Precision Contour Control Parameters related to Service ...

and other parameters

Quite a few parameters have nothing to do with daily programming and are listed only as an actual example. All system parameters should be set or changed only by a qualified person, such as an experienced service technician. A programmer or operator should not modify any parameter settings. These changes require not only qualifications but authorization *as well*. Keep the list of original parameter settings away from the control, in a safe place, just in case.

Take care when changing control system parameters!

Many parameters are periodically updated during program processing. CNC operator is usually not aware that this activity is going on at all. There is no real need to monitor this activity. The safest rule to observe is that once the parameters have been set by a qualified technician, any *temporary* changes required for a given work should be done through the CNC program. If *permanent* changes are required, an authorized person should be assigned to do them - *nobody else*. Some parameters may be changed **very carefully** - through the program.

System Defaults

Many parameter settings stored in control at the time of purchase have been entered by the manufacturer as either the *only choices*, the *most suitable choices*, or the *most common selections*. That does not mean they will be *the* preferred settings - it means they were selected based on their common usage. Many settings are rather conservative in their values, for safety reasons.

The set of parameter values established at the time of installation are called the *default* settings.

The English word '*default*' is a derivative of a French word '*defaut*', that can be translated as '*assumed*'. When main power to the control is turned on, there are no set values passed to parameters from a program, since no program has yet been used. However, certain settings become active automatically, *without* an external program. For instance, a cutter radius offset is automatically canceled at the control system startup. Also canceled are the fixed cycle mode and tool length offset. The control '*assumes*' that certain conditions are preferable to others. Many operators will agree with most of these initial settings, although not necessarily with all of them. Some settings are customizable by a change of a parameter settings. Such settings will become permanent and create a *new 'default'*.

Always document any changes made to parameters!

A computer is fast and accurate but has no intelligence. People are often slow and make errors but have one unique ability - *they think*. A computer is just a machine that does not *assume* anything, does not *consider*, does not *feel* computer *does not think*. A computer does not do anything that a human effort and ingenuity has not done during the design process, in form of hardware and software.

When a CNC machine is powered, its internal software sets *certain* existing parameters to their default condition, designed by engineers. Not *all* system parameters, only *certain* parameters can have an assumed condition - a condition that is known as the *default value* (condition).

For example, a tool motion has three basic modes - a *rapid* motion, a *linear* motion and a *circular* motion. The default motion setting is controlled by a parameter. Only one setting can be active at the startup. Which one? The answer *depends on the parameter setting*. Many parameters can be preset to a desired state. Only the *rapid* or *linear* mode can be set as default in the example. Since rapid motion is the first motion in most programs, it seems to make sense to make it a default - *but wait!*

Most controls are set to *linear* motion as the default (G01 command) to be in effect at the start - *strictly for safety reasons*. When machine axes are moved manually, the parameter setting has no effect. If a manual input of an axis command value takes place, either through the program or from the control panel, a tool motion results. If a motion command is not specified, CNC system will use the command mode that had been preset as the default in the parameters. Since the default mode is linear motion G01, the result is an *error condition*, faulting the system *for the lack of feed rate!* There is no cutting feed rate in effect, which the G01 requires. Had the default setting been rapid motion G00, a rapid motion would be performed, as it does not need programmed feed rate.

It is beneficial to know the default settings of all controls in the shop. Unless there is a good reason to do otherwise, defaults for similar controls should be the same.

Memory Capacity

CNC programs can be stored in the control memory. Program size is only limited by the control internal capacity. This capacity is measured in variety of ways, originally as the equivalent *length of tape* in meters or feet, lately as the *number of bytes* or the number of *screen pages*. A common minimum memory capacity of a small CNC lathe control may be 20m of tape (66 ft). This is an old-fashioned method that somehow persisted in staying with us. On CNC milling systems, the memory requirements based on the same criteria are generally larger and the typical minimum memory capacity is 80 m or about 263 ft. Optionally, larger memory capacity can be added to any control system. The minimum memory capacity of a control varies from one machine to another - always check control specifications carefully.

Modern methods of measuring memory capacity prefer to use *bytes* as the unit, rather than a length of an obsolete tape. A byte is the smallest unit of storage capacity and is very roughly equivalent to one character in the program.

Memory capacity of the control system should be large enough to store the longest CNC program expected on a regular basis. That requires some planning before the CNC machine is

purchased. For example, in three-dimensional mold work or high-speed machining, the cost of additional memory capacity may be very high. Although any cost is a relative term, there are reliable and inexpensive alternatives, well worth looking into.

One alternative is running the CNC program from a personal computer. An inexpensive communication software and cabling is required to connect the computer with the CNC system. The simplest version is to *transfer* a CNC program from one computer to the other. More sophisticated possibility includes software and cables that can run the machine from a personal computer, without loading it into the CNC memory first. This method is often called '*drip-feeding*' or '*bitwise input*'. When operated from a personal computer, the CNC program can be as long as the storage device capacity, typically the hard drive.

Most CNC programs will fit into the CNC internal memory. Latest Fanuc controls show the available memory as the number of *free screen display pages*.

In cases where the available memory capacity is too small to accept large programs, several techniques are available to minimize the problem, for example, eliminating leading zeros or removing all or some block numbers.

C. MANUAL PROGRAM INTERRUPTION

If a program needs to be interrupted in the middle of processing, control system offers several ways to do that, using the machine operation panel. The most common features of this type are toggle switches or push buttons for a *single block* operation, *feed hold* and *emergency stop*.

Single Block Operation

The normal purpose of a program is to control the machine tool automatically and sequentially in a continuous mode. Every program is a series of formatted commands or instructions - written as individual lines of code, called *blocks*. Blocks and their concepts will be described in the following chapters. All program commands in a single block are processed as a *single instruction*. The blocks are received by the control system in sequential order, from top down and in the order, they appear in the program. Normally, a CNC machine is run in a continuous mode, while all blocks are processed automatically, one after another. This continuity is important for production, but not practical when proving a new program, for example.

To disable continuous program execution, a *Single Block* switch is provided on the operation panel. In the single block mode, only one program block will be processed each time the *Cycle Start* key is pressed. On the operation panel, single block mode can be used separately or in combination with other settings that make program proving faster and more accurate.

Feed hold

Feed hold is a special push button located on the operation panel, usually close to the *Cycle Start* button. When this button is pressed during a rapid, linear or circular axes motion, it will immediately stop the motion. Such action applies to all axes active at the time. This feature is convenient for machine setup or first part run. Some types of motion restrict the Feed hold function or disable it altogether. For example, threading or tapping modes make the switch inoperative.

Activating Feed hold at the machine will not change any other program values - it will only affect axis motion. Feed hold switch will be illuminated (in red light), as long as it is effective. CNC programmer can override Feed hold from *within* the program, for special purposes.

Emergency Stop

Every CNC machine has at least one special mushroom shaped push button, red in color, which is in an accessible place on the machine. It is marked as the *Emergency Stop* or *E-stop*. When this button is pressed, *all machine activities will cease immediately*. The main power supply will be interrupted, and the machine will have to be restarted. Emergency stop switch is a mandatory safety feature on all CNC machines.

Pressing the emergency stop button is not always the *best* or even the *only* way to stop a machine operation. In fact, the latest controls offer other features, far less severe, designed to prevent a collision between cutting tool and part or fixture. Previously discussed Feed hold button is only one option, along with several other features. If the emergency stop must be used at all, it should be used only as the *last resort*, when any other action would require unacceptably longer time. There is no need for panic, if something does go wrong.

For some machine actions, the effect of *Emergency Stop* is not always apparent. For example, the spindle requires a certain time for deceleration to stop.

D. MANUAL DATA INPUT - MDI

ACNC machine is not always operated by the means of a program. During part setup, the CNC operator has to do a number of operations that require physical movements of machine slides, rotation of spindle, tool change, etc. There are no mechanical devices on a CNC machine. The handle (*Manual Pulse Generator*) is an electronic, not a mechanical unit. In order to operate a CNC machine without conventional mechanical devices the control system offers a feature called *Manual Data Input* - or MDI.

Manual Data Input enables the input of program data into the system *one program instruction* at a time. If too many instructions were to be input repeatedly, such as a long program, the procedure would be very inefficient. During setup and for similar purposes, one or a few instructions at a time will benefit from MDI.

To access MDI mode, the MDI key on the operation panel must be selected. That opens the screen display with the current system status. Not all, but the majority of programming codes are allowed in the MDI mode. Their format is identical to the format of a CNC program in written form. This is one area where CNC operator acts as a CNC programmer. It is very important that the operator is trained at least in the basics of CNC programming, certainly to the point of being able to handle the setup instructions for *Manual Data Input*.

E. PROGRAM DATA OVERRIDE

All CNC units are designed with a number of special rotary switches that share one common feature - they allow the CNC operator to *override* programmed spindle speed or programmed speed of an axis motion. For example, a 15 in/min feed rate in the program produces a slight chatter. A knowledgeable operator will know that by *increasing* the feed rate or *decreasing* the spindle speed, the chatter may be eliminated. It is possible to change the feed rate or the spindle speed by editing the program, but this method is not very efficient. A certain 'experimentation' may be necessary during actual cut to find the optimum setting value. Manual override switches come to the rescue because they can be used by trial during operation. There are four override switches found on most control panels:

Rapid feed rate override (rapid traverse)	modifies the rapid motion of a machine tool
Spindle speed override	modifies the programmed spindle r/min
Feed rate override (cutting feed rate)	modifies the programmed feed rate
Dry run mode	changes cutting motions to a variable speed

Override switches can be used individually or together. They are available on the control to make work easier for both the operator and the programmer. Operator does not need to 'experiment' with speeds and feeds by constantly editing the program and the programmer has a certain latitude in setting reasonable values for cutting feed rates and spindle speed. The presence of override switches is not a license to program unreasonable cutting values. Overrides are fine tuning tools only - part program must always reflect the machining conditions of the work. Usage of override switches does not make any program changes but gives the CNC operator an opportunity to edit

the program later to reflect all optimum cutting conditions. Used properly, override switches can save a great amount of valuable programming time as well as setup time at CNC machine.

Rapid Motion Override

Rapid motions are selected in the CNC program by a preparatory command without a specified feed rate. If a machine is designed to move at 985 in/min (25000 mm/min) in the rapid mode, this rate will never appear in the program. Instead, the rapid motion mode is called by programming a special preparatory command G00. During program processing, all motions in G00 mode will be at the manufacturer's fixed rate. The same program will run faster on a machine with a high rapid motion rating than on a machine with a low rapid motion rating.

During setup, the rapid motion rate may require some control for program proving, when very high rapid rates are uncomfortable to work with. After a program had been proven, rapid rate can be applied at its maximum. CNC machines are equipped with a *rapid override switch* to allow temporary rapid motion settings. Located on the machine control panel, this switch can be set to one of four settings. Three of them are usually marked as the *percentage* of the maximum rate, typically as 100%, 50% and 25%. By switching to one of them, the rapid motion rate changes. For example, if the maximum rapid rate is 985 in/min or 25000 mm/min, the actual reduced rates are 493 in/min or 12500 mm/min at the 50% setting and 246 in/min or 6250 mm/min at the 25% setting. Each of the reduced rates is more comfortable to work with during setup.

The fourth position of the switch often has no percentage assigned and is identified as an F1 or by a small symbol. In this setting, the rapid motion rate is even slower than that of 25% setting. Why is it not identified as 10% or 15%, for example? The reason is simple - control system allows a *customized* selection as to what the setting will be. It may be a setting of between 0 and 100%. The default setting is also the most logical - usually 10% of the maximum rapid traverse rate. This setting should never be higher than 25% and can be done only through setting of a system parameter. Make sure that all persons who work on such a machine are aware of the changes.

Spindle Speed Override

The same logic used for the application of rapid rate override can be used for the spindle speed override. The required change can be established during actual cutting by using the *spindle speed override switch*, located on the machine control panel. For example, if the programmed spindle speed of 1000 r/min is too high or too low, it may be changed temporarily by the switch. During actual cutting, the CNC operator may experiment with the spindle speed override switch to find the optimum speed for given cutting conditions. This method is a much faster than 'experimenting' with the program values.

Spindle speed override switch can be continuous on some controls or selectable in increments of 10%, typically within the range of 50-120% of the programmed spindle speed. A spindle programmed at 1000 r/min can be overridden during machining to 500, 600, 700, 800, 900, 1000, 1100 and 1200 r/min. This large range allows the CNC operator certain flexibility of optimizing spindle rotation to suit given cutting conditions. There is a catch, however. The optimized spindle speed change may apply to only one tool of the many often used in the program. No CNC operator can be expected to watch for that particular tool and switch the speed up or down when needed. A simple human oversight may ruin the part, the cutting tool or both. The recommended method is to find out the optimum speed for each tool, write it down, then change the program accordingly, so *all tools* can be used at the 100% spindle override setting for production.

Comparison of the increments on the spindle override switch with the increments on switches for rapid traverse override (described earlier) and the feed rate override (described next), offers much more limited range. The reason for the spindle speed range of 50% to 120% is safety. To illustrate with a rather exaggerated example, no operator would want to mill, drill or cut any material at 0 r/min (no spindle rotation), possibly combined with a heavy feed rate.

To change the selected override setting into 100% speed in the program, a new spindle speed

must be calculated. If a programmed spindle speed of 1200 r/min for a tool is *always* set to 80%, it should be edited in the program to 960 r/min, then used at 100%. The formula is quite simple:

$$S_n = \frac{S_p \times p}{100}$$

Where:

S_n = *Optimized-or NEW r/min*

S_p = *Originally programmed r/min*

p = *Percentage of spindle override*

Overriding the programmed spindle speed on CNC machines should have only one purpose - to establish spindle speed rotation for the best cutting conditions.

Feed rate Override

Probably the most used override switch is one that changes programmed feed rates. For milling controls, the feed rate is programmed in *in/min* or *m/min*. For lathe controls, the feed rate is programmed in *in/rev* or in *mm/rev*. Feed rate per minute on lathes is used only in cases when the spindle is not rotating, and the feed rate needs to be controlled.

The new feed rate calculation, based on the overridden feed rate setting, is similar to that for spindle speed:

$$F_n = \frac{F_p \times p}{100}$$

Where:

F_n = *Optimized - or NEW - feed rate*

F_p = *Originally programmed feed rate*

p = *Percentage of feed rate override*

Feed rate can be overridden within a large range, typically from 0% to 200% or at least 0% to 150%. When the feed rate override switch is set to 0%, the CNC machine will *stop* the cutting motion. Some CNC machines do not have the 0% percent setting and start at 10%. This can be change by a system parameter. The maximum of 150% or 200% cutting feed rate will cut 1.5 or 2 faster than the programmed feed rate amount.

There are situations, where the use of a feed rate override would damage the part or the cutting tool - or both. Typical examples are various tapping cycles and single point threading. These operations require spindle rotation synchronized with the feed rate. In such cases, the feed rate override will become *ineffective*. Feed rate override *will* be effective, if standard motion commands G00 and G01 are used to program any tapping or thread cutting motions. Single point threading command G32, tapping fixed cycles G74 and G84, as well as lathe threading cycles G92 and G76 have feed rate override cancellation built into the software. All these and other related commands are described later in the digital book, in more detail.

Dry Run Operation

Dry run is a special kind of override. It is activated from the control panel by the *Dry Run* switch. It only has a direct effect on the feed rate and allows much higher feed rate than that used for actual machining. In practice, it means the program can be executed much faster than using a feed rate override at the maximum setting. No actual machining takes place when the dry run switch is in effect.

What is the purpose of the dry run and what are its benefits? Its purpose is to test the program integrity *before* the CNC operator cuts the first part. Benefits are mainly in the time saved during

program proving when no machining takes place. During dry run, part is normally *not* mounted in the machine. If the part is mounted in a holding device and the dry run is used as well, it is very important to provide sufficient clearances. Usually, it means *moving the tool away from the part*. Program is then processed 'dry', without actual cutting, without a coolant, just in the air. Because of the heavy feed rates in dry run, no part can be machined safely. During dry run, the program can be checked for all possible errors except those that relate to the actual contact of cutting tool with the material.

Dry run is a very efficient setup aid to prove overall integrity of a CNC program. Once a program is proven during dry run, CNC operator can concentrate on those sections of the program that contain actual machining. Dry run can be used in combination with several other features of the operation panel.

Make sure to disable dry run before machining!

Z-Axis Neglect

Another very useful tool for testing unproven programs on CNC machining centers (*not* lathes) is a toggle switch located on the operation panel called *Z-axis Neglect* or *Z-axis Ignore*. As either name suggests, when this switch is activated, any motion programmed for the Z-axis will *not* be performed. Why the Z-axis? Since the X and Y axes are used to profile a part shape (the most common contouring operations), it would make no sense to temporarily cancel either one of these axes. By temporarily neglecting, that is *disabling*, the Z-axis temporarily, CNC operator can concentrate on proving the accuracy of the part contour, without worrying about depth motions. Needless to say, this method of program testing must take place without a mounted part, and normally without a coolant as well. *Be careful here!* It is important to *enable* or *disable* the switch *at the right time*. If the Z-axis motion is disabled before the *Cycle Start* key is pressed, all following Z-axis commands will be ignored. If the motion is enabled or disabled during program processing, the position of Z-axis may be inaccurate.

Z-axis neglect switch may be used in both manual and automatic modes of operation. Just make sure that the motion along the Z-axis is changed back to the *enabled* mode, once the program proving is completed. Some CNC machines require resetting of the Z-axis position settings.

Manual Absolute Setting

Some older CNC machines had a toggle switch identified as *Manual Absolute*, which could be set to ON or OFF position. If installed, its purpose is simple - if a manual motion is made during program processing, for example to move a drill to inspect a hole, work coordinates are updated if the switch is ON, but they are *not* updated if the switch is OFF. In practice, this switch should always be ON - and for that reason, most controls do not have this switch anymore.

Sequence Return

Sequence Return is a special function controlled by a switch or a key on the control panel. Its purpose is to enable the CNC operator to start a program from the middle of an interrupted program. Certain programmed functions are memorized (usually the last speed and feed), others have to be input by the *Manual Data Input* key. Operation of this function is closely tied to actual machine tool design. More information on the usage can be found in the machine tool manual. This function is very handy when a tool breaks during processing of very long programs. It can save valuable production time, if used properly.

Auxiliary Functions Lock

There are three functions available to the operation of a CNC machine that are part of '*auxiliary functions*' group. These functions are:

Miscellaneous functions lock	Locks 'M' functions
Spindle functions lock	Locks 'S' functions

Tool functions lock	Locks 'T' functions

As described later in this chapter, auxiliary functions generally relate to *technological* aspects of CNC programming. They control such machine functions as spindle rotation, spindle orientation, coolant selection, tool changing, indexing table, pallets and many others. To a lesser degree, they also control some program functions, such as compulsory or optional program stop, subprogram flow, program closing and others.

When auxiliary functions are locked, all machine related miscellaneous functions M, all spindle functions S and all tool functions T will be temporarily suspended. Some machine manufacturers prefer the name *MST Lock* rather than *Auxiliary Functions Lock*. MST is an acronym of the first letters from the words *Miscellaneous*, *Spindle* and *Tool*, referring to the program functions that will be locked.

Applications of these locking functions are limited to job setup and program proving only and are not used for production machining.

Machine Lock

Machine Lock function is yet another control feature for program proving. So far, we have looked at the *Z-axis Neglect* function and locking of the auxiliary functions. Remember that the *Z-axis Neglect* function will disable the motion of Z-axis only and the *Auxiliary Functions Lock* (also known as *MST lock*) locks miscellaneous functions, spindle functions and tool functions. Another function, also available through the control panel, is called *Machine Lock*. When this function is enabled, motion of *all* axes is locked.

It may seem strange to test a program by locking all tool motions, but there is a good reason to use this feature. It gives the CNC operator a chance to test the program with virtually no chance of a collision.

When machine lock is enabled, *only the axis motion is locked*. All other program functions are available, including tool change and spindle functions. This function can be used alone or in combination with other functions in order to discover possible program errors. The most typical errors are syntax errors and the various tool offset functions.

Practical Applications

Many of the control features described in this chapter, are used in *conjunction* with each other. A good example is *Dry Run* used in conjunction with the *Z-axis Neglect* or the *Auxiliary Functions Lock*. By knowing what function are available, CNC operator makes a choice to suit the needs of the moment. There are many areas of equal importance on which the CNC operator has to concentrate when setting up a new job or running a new program. Many features of the control unit are designed to make operator's job easier. They allow a focus on one or two items at a time rather than the complexity of the whole program. These features have been covered in a reasonable detail, now is the time to look at some practical applications.

During initialization of a new program run, a good CNC operator will take certain precautions as a matter of fact. For example, the first part of the job will most likely be tested with a rapid motion set to 25% or 50% of the available rapid rate. This reduced setting allows the operator to monitor the program integrity, as well as specific details. These details may include items such as a possibility of insufficient clearance between tool and stock, checking if the toolpath looks reasonable, and so on.

CNC operator will have a number of tasks to perform simultaneously. Some of these tasks include monitoring spindle speed, feed rate, tool motions, tool changes, coolant, etc. A careful and conscious approach results in building the confidence in the integrity of a CNC program. It may be the second or even the third part of the job when CNC operator starts thinking of the optimization of cutting values, such as spindle speed and cutting feed rates. This optimization will

truly reflect the ideal speeds and feeds for a particular part under given setup.

Production supervisor should not arbitrarily criticize an override setting less than 100%. Many managers consider the CNC program as an unchangeable and perfect document. They take the attitude that what is written is infallible - which is not always true. Often, the CNC operator may have no other choice *but to override* programmed values. What is most important, is modification of the program that reflects the optimized cutting conditions.

Once the machine operator finds what values must be changed in the program itself, this program must be edited to reflect these changes. Not only for the job currently worked on, but also for any repetition of the same job in the future. After all, it should be the goal of every programmer and CNC operator, to run any job at one hundred percent efficiency. This efficiency is most likely reached as a combined effort of the operator and the programmer. Good CNC programmer will always make the *effort* to reach 100% efficiency at the desk and then improve the program even further.

F. SYSTEM OPTIONS

Optional features on a CNC system are like options on a car. What is an option at one dealership, maybe a standard feature at another. Marketing strategies and corporate philosophies have a lot to do with this approach.

Here is a look at some control features that may or may not be classified as optional on a particular system. But some important disclaimer first:

This digital book covers subject matter relating to the majority of control features, regardless of whether they are sold as a standard or an optional feature of the system.

It is up to the user to find out what exact options are installed on a particular control system.

Graphic Display

Graphic representation of toolpath on the display screen is one of the most important, as well as sought after, control options. Do not confuse this option with any type of conversational programming, which also uses a graphic toolpath interface. In the absence of a computer assisted programming (CAM), graphic display on the control panel is a major benefit. Whether in monochrome or color, the convenience of seeing the tool motions *before* actual machining is much appreciated by CNC operators and programmers alike.

A typical graphics option shows machine axes and two cursors for zooming. When the toolpath is tested, individual tools are distinguished by different colors, if available, or different intensity. Rapid motions are represented by a dashed line type, cutting motions by a continuous line type. If the graphics function is applied during machining, tool motions can be watched on the display screen - very helpful for those CNC machines that have dirty, oily and often scratched safety shields.

Upwards or downwards scaling of the display allows for evaluation of a tool motion overall or for detail areas. Many controls also include actual toolpath simulation, where the part shape and the cutting tool can be set first, then seen on the screen.

In-Process Gauging

During many unattended machining operations, such as in manufacturing cells or Agile manufacturing, a periodic checking and adjusting dimensional tolerances of the part is imperative. As the cutting tool wears out, or perhaps because of other causes, the dimensions may fall into the 'out-of-tolerance' zone. Using a probe device and a suitable program, *In-Process Gauging* option offers quite a satisfactory solution. CNC part program for the *In-Process Gauging* option will contain some quite unique format features - it will be written parametrically and will be using another option of the control system - the *Custom Macros* (sometimes called the *User Macros*), which offer variable and parametric type programming.

If a company or machine shop is a user of *In-Process Gauging* option, there are good chances that other control options are also installed and available to the CNC programmer. Some of the most typical options are probing software, tool life management, macros, etc. This technology goes a little too far beyond standard CNC programming, although it is closely related and frequently used. Companies that already use the numerical control technology, will be well advised to look into these options to remain competitive in their field.

Stored Stroke Limits

Definition of an area on CNC lathes or a cube on CNC machining centers that is safe to work within, can be stored as a control system parameter called *stored stroke limit*. These stored stroke limits are designed to prevent a collision between the cutting tool and a fixture, machine tool or part. The area (2D) or the cube (3D) can be defined as either *enabled* for the cutter entry or *disabled* for the cutter entry. It can be set manually on the machine or, if available, by a program input. Some controls allow only one area or cube to be defined, others allow more.

When this option is in effect and the CNC unit detects a motion in the program that takes place within the forbidden zone, an error condition results, and the machining is interrupted. A typical application may include zones occupied by a tailstock, a fixture, a chuck, a rotary table, and even an unusually shaped part.

Drawing Dimensions Input

An option that seems somewhat neglected, is the programming method by using input of dimensions from an engineering drawing. The ability to input known coordinates, radiuses, chamfers and given angles directly from the drawing makes it an attractive option. This ability is somewhat overshadowed by poor program portability. Such an option must be installed on all machines in the shop, in order to use the programmed features efficiently. This is not a common feature and is not covered in this digital book.

Machining Cycles

Both milling and turning controls offer a variety of machining cycles. Typical machining cycles for milling operations are called *fixed cycles*, also known as *canned cycles*. They simplify simple point-to-point machining operations such as drilling, reaming, boring, back boring and tapping. Some CNC systems also offer cycles for face milling, pocket milling, various hole patterns, etc.

CNC lathes also have many machining cycles available to remove material by automatic roughing, profile finishing, facing, taper cutting, grooving and threading. Fanuc controls call these cycles *Multiple Repetitive Cycles*.

All these cycles are designed for easier programming and faster changes at the machine. They are *built in* the control and *cannot be changed*. Programmer supplies the cutting values during program preparation by using an appropriate cycle call command. All processing is done automatically, by the CNC system. Of course, there will always be special programming projects that cannot use any cycles, at least not effectively, and have to be programmed manually or with the use of an external computer and CAM software.

Cutting Tool Animation

Many of the graphic toolpath displays defined earlier, are represented by simple lines and arcs. Current tool position is usually the location of a line or arc endpoint on the screen. Although this method of displaying cutting tool motion graphically is certainly useful, there are two disadvantages to it. The cutting tool shape and the material being removed cannot be seen on the screen, although toolpath simulation may help a bit. Many modern controls incorporate graphic feature called *Cutting Tool Animation*. If available on the control, it shows the part blank, the mounting device and the tool shape. As the program is processed, CNC operator has a reasonably accurate visual aid in program proving. Each graphic element is identified by a different color, for even a better appearance. The blank size, mounting device and tool shape can be preset for exact proportions and a variety of tool shapes can be stored for repetitive use. This option is a good

example of CAD/CAM like features built into a stand-alone control system.

Connection to External Devices

CNC computer (control) can be connected to an external device, usually another computer. Every CNC unit has one or more connectors, specifically designed for interfacing to peripheral devices. The most common device is called RS-232 (EIA standard), designed for communications between two computers. Setting up the connection with external devices is a specialized application. CNC operator uses such a connection to transfer programs and other settings between two computers, usually for storage and backup purposes. Devices other than RS-232 are also available - check with the machine vendor.

4. CNC TURNING

Conventional engine lathe or a turret lathe is a common machine in just about every machine shop. A lathe is used for machining cylindrical or conical work, such as shafts, rings, wheels, bores, threads, etc. The most common lathe operation is removal of material from a round stock, using a turning tool for external cutting. A lathe can also be used for internal operations such as boring, as well as for facing, grooving, threading, etc., if a proper cutting tool is used. Turret lathes are usually weaker in machining power than engine lathes, but they do have a special holder that stores several mounted cutting tools. An engine lathe has often only one or two cutting tools mounted at a time but has more machining power.

Typical lathe work controlled by a CNC system uses machines known in industry as the *CNC Turning Centers* - or more commonly - *CNC lathes*.

The term '*turning center*' is rather unpopular, but an accurate overall description of a computerized lathe (a CNC lathe) that can be used for a great number of machining operations during single setup. For example, in addition to the standard lathe operations such as turning and boring, CNC lathe can be used for drilling, grooving, threading, knurling and even burnishing. It can also be used in different modes, such as chuck and collet work, bar feeder, or between centers. Many other combinations also exist. CNC lathes are designed to hold several tools in special turrets, they can have a milling attachment (live tooling), indexable chuck, a sub-spindle, tailstock, steady rest and many other features not always associated with a conventional lathe design. Lathes with more than four axes are also common. With constant advances in machine tool technologies, more CNC lathes appear on the market that are designed to do a number of operations in a single setup, many of them traditionally reserved for a mill or machining center.

A. TYPES OF CNC LATHES

Basically, CNC lathes can be categorized by the *type of design* and by the *number of axes*. The two basic types are a *vertical* CNC lathe and a *horizontal* CNC lathe. Of the two, the horizontal type is by far the most common in manufacturing and machine shops. A vertical CNC lathe (incorrectly called a vertical boring mill) is somewhat less common but is irreplaceable for a large diameter work. For CNC programmer, there are no significant differences in programming approach between the two lathe types.

B. NUMBER OF AXES

The most common distinction of different CNC lathes is by the number of programmable axes. Vertical CNC lathes have two axes in most designs available. The much more common CNC horizontal lathes, commonly designed with two programmable axes, are also available with three, four, six, and more axes, thus adding extra flexibility to manufacturing of more complex parts.

A typical horizontal CNC lathe can further be described by the *type* of engineering design:

FRONT lathe: an engine lathe type

REAR lathe: a unique slant bed type

Slant bed type is very popular for general work, because its design allows cutting chips to fall away from the CNC operator and, in case of an accident, forces the part to fall down into a safe area, towards chip conveyer.

Between the categories of flat bed and slant type lathes, front and rear lathes, horizontal and vertical lathe designs, there is another variety of a lathe. This category describes CNC lathes by the *number of axes*, which is probably the simplest and most common method of lathe identification.

C. AXES DESIGNATION

A typical CNC lathe is designed with two standard axes one axis is the X-axis, the other axis is the Z-axis. Both axes are perpendicular to each other and represent the typical two-axis lathe motions. X-axis also represents *cross travel* of the cutting tool; Z-axis represents its *longitudinal motion*. All varieties of cutting tools are mounted in a turret (a special tool magazine) and can be external or internal. Because of this design, a turret loaded with all cutting tools moves along both X and Z axes, which means all tools are in the work area at all times.

Following the established standards of milling machines and machining centers, the only machine axis capable of making a hole by methods of drilling, boring, piercing or punching, is the Z-axis.

In CNC lathe work, the traditional axis orientation for a horizontal type of lathe is *upwards* and *downwards* motion for the X-axis, and *left* and *right* motion for the Z-axis, when looking from the machinist's position. This view is shown in the following three illustrations *Figure 3-1*, *Figure 3-2*, and *Figure 3-3*.

11

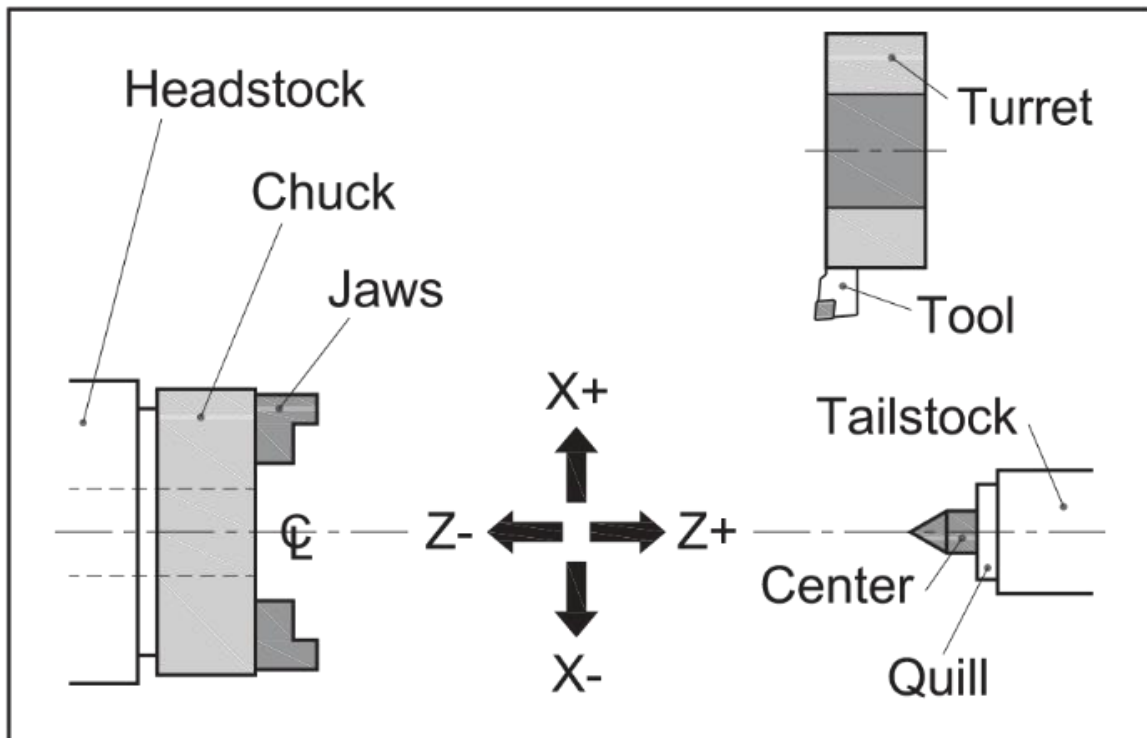


Figure 3-1

Typical configuration of a two-axis slant bed CNC lathe - rear type

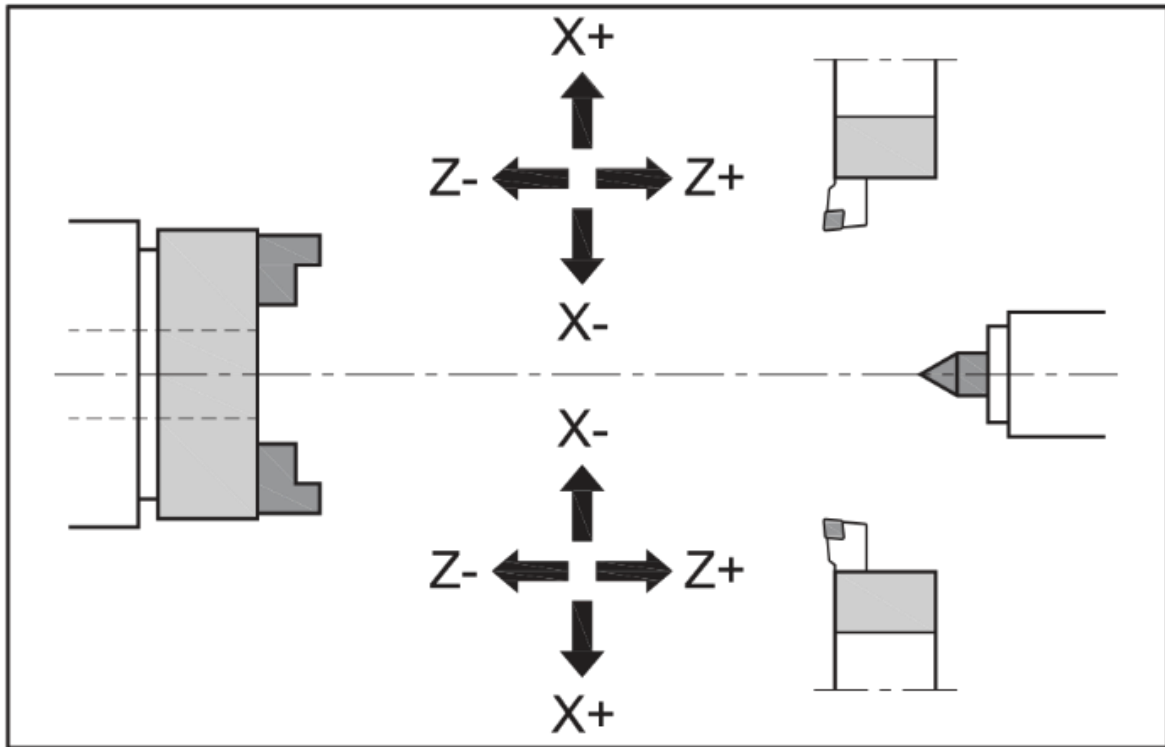


Figure 3-2

Typical configuration of a CNC lathe with two turrets

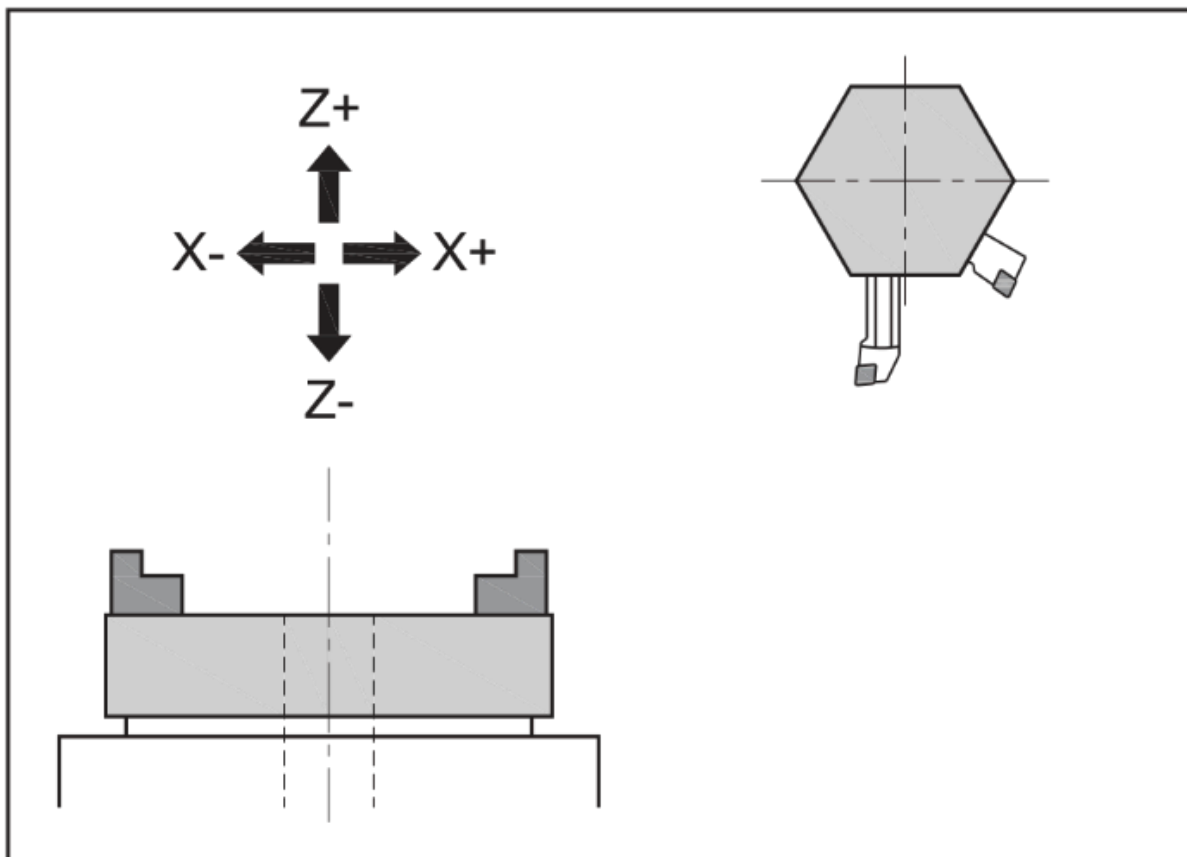


Figure 3-3

Schematic representation of a vertical CNC lathe

This is true for both front and rear lathes, and for lathes with three or more axes. Chuck face is oriented vertically to the horizontal spindle center line for all horizontal lathes. Vertical lathes, due to their design, are rotated by 90, where the chuck face is oriented horizontally to the vertical

spindle center line.

In addition to the X and Z primary axes, the multi-axis lathes have individual descriptions of each additional axis, for example, the C-axis or Y-axis, used for milling operations, using the so called *live tooling*. More details on the subject of coordinate system and machine geometry are available in the next chapter.

Two-axis Lathe

This is the most common type of CNC lathes. The work holding device, usually a chuck, is mounted on the left side of the machine (as viewed by the operator). Rear type, with slant bed, is the most popular design for general work. For some special work, for example in the petroleum industry (where turning tube ends is a common work), a flatbed is usually more suitable. Cutting tools are held in a specially designed indexing turret that can hold four, six, eight, ten, twelve and more tools. Many such lathes also have two turrets, one on each side of the spindle center line.

Advanced machine tool designs incorporate tool storage away from the work area, similar to design of machining centers. Tens and even hundreds of cutting tools may be stored and used for a single CNC program. Many lathes also incorporate a quick changing tooling system.

Three-axis Lathe

Three-axis lathe is essentially a two-axis lathe with an additional axis. This axis has its own designation, usually as a C-axis in absolute mode (H-axis in incremental mode) and is fully programmable. Normally, the third axis is used for cross-milling operations, slot cutting, bolt circle holes drilling, hex faces, side faces, helical slots, etc. This axis can replace some simple operations on a milling machine, reducing the setup time for the job. Some limitations do apply to many models, for example, the milling or drilling operations can take place only at positions projecting from the tool center line to the spindle center line (within a machining plane), although others offer off-center adjustments.

The third axis has its own power source, but the power rating is relatively lower when compared with the majority of machining centers. Another limitation may be the smallest increment of the third axis, particularly on the early three axis lathes. Smallest increment of one degree is certainly more useful than an increment of two or five degrees. Even better is an increment of 0.1, 0.01, and commonly

0.001 on the latest models. Usually, lathes with three axes offer a very fine radial increment that allows a simultaneous rotary motion. Those with low increment values are usually designed with an oriented spindle stop only.

From the perspective of CNC part programming, the additional knowledge required is a subject not difficult to learn. General principles of milling apply and many programming features are also available, for example, fixed cycles and other shortcuts.

Four-axis Lathe

By design, a four-axis CNC lathe is a totally different concept than a three-axis lathe. As a matter of fact, to program a four-axis lathe is nothing more than programming *two two-axis lathes at the same time*. That may sound strange at first, until the principle of a four-axis CNC lathe becomes clearer.

There are actually two controls and two sets of XZ axes, one for each pair (set) of axes. Only one program may be used to do the *external* - or *outside* - diameter roughing (OD) and another program to do the *inside* - or *internal* roughing (ID). Since a four-axis lathe can work with each pair of axes *independently*, the OD and ID can be machined at the same time, doing two different operations simultaneously. The main keys to a successful 4-axis lathe programming are coordination of the tools and their operations, timing of the tool motions and a generous sense of healthy compromise.

For several reasons, both pairs of axes cannot work all the time. Because of this restriction,

special programming features such as synchronized waiting codes (typically *Miscellaneous Function*), the ability to estimate how much time each tool requires to complete each operation, etc., are required. There is a level of compromise here, because only *one* spindle speed can be used for both active cutting tools, although feed rate is independent for both pairs of axes. This means that some machining operations simply cannot be done simultaneously.

Not every lathe job benefits from 4-axis machining. There are cases when it is more costly to run a job on a 4-axis lathe inefficiently and it may be very efficient to run the same job on a 2-axis CNC lathe.

Six-axis Lathe

Six-axis CNC lathes are specially designed lathes with a twin turret and a set of three axes per turret. This design incorporates many tool stations, many of them power driven, as well as back-machining capabilities. Programming these lathes is similar to programming a three-axis lathe twice. Control system automatically provides synchronization, when necessary.

A small to medium size six-axis CNC lathe is popular choice of screw machine shops and industries with similar small parts and large volume applications.

D. FEATURES AND SPECIFICATIONS

A look at a typical promotional brochure describing a CNC machine tool is very useful in many respects. In most cases, the artwork quality is impressive, the printing, photographs, paper selection and the use of colors is equally well done. It is the purpose of the brochure to make a good marketing tool and attract the potential buyer.

However, there is more in a promotional brochure than just attractive illustrations - in fact, a well-designed brochure offers a wealth of technical information, describing the machine tool in many details. These are *features* and *specifications* the CNC machine tool manufacturer considers important to the customer.

In the majority of brochures, there are practical data that can be used in programming a particular CNC machine, a lathe in the example.

Typical Machine Specifications

A typical horizontal CNC lathe, with two axes and a slant bed design, may have the following specifications (taken from an actual brochure):

Description	Specification
Number of axes	Two (X, Z) or three (X, Z and C)
Maximum swing over bed	560 mm - 22.05 inches
Maximum turning diameter	350 mm - 13.76 inches
Maximum turning length	550 mm - 21.65 inches
Spindle bore	85 mm - 3.34 inches
Bar capacity	71 mm - 2.79 inches
Number of tools	12
Tool size square	25 mm - 1 inch
Tool size round	40 mm - 1.57 inches
Indexing time	0.15 second

Axis travel in X-axis	222 mm - 8.75 inches
Axis travel in Z-axis	635 mm - 25 inches
Rapid traverse rate X-axis	16000 mm/min 629 in/min
Rapid traverse rate Z-axis	24000 mm/min 944 in/min
Cutting feed rate	0.01 - 500 mm/rev 0.0001 - 19.68 in/rev
Chuck size	254 mm - 10 inches
Main spindle motor	AC 15/11kW AC 20/14.7HP
Spindle speed	35 - 3500 rpm
Minimum input increment	0.001 mm - 0.0001 inch
Motorized head:	
Number of rotating tools	12
Rotating tool speed	30 - 3600 r/min
Milling motor	AC 3.7/2.2 kW AC 5/2.95 HP
Collet size	1 - 16 mm - 0.04 - 0.63 inches
Tap size	M3 - M16 metric #5 - 5/8 inches

It is very important to understand all specifications and features of the CNC machine tools in the shop. Many features relate to the control system, many others to the machine tool itself. In CNC programming, many important decisions are based on one or several of these features, for example number of tool stations available, maximum spindle speed and others.

Control Features

The last item in understanding the overall description of a CNC lathe is a look at some control features unique to lathes and how they differ from a typical milling control.

At this time, some features and program codes may not make much sense - they are included for reference only. Common and typical features are listed:

X-axis represents a diameter, not a radius

Constant surface speed (CSS) - also known as cutting speed (CS) - is standard control feature (G96 for cutting speed and G97 for spindle speed)

Absolute programming mode is X or Z or C

Incremental programming mode is U or W or H

Thread cutting of various forms (including taper and circular) can be performed, depending on the control model

Dwell can use the P, U or X address (G04)

Tool selection uses 4-digit identification

Feed rate selection (normal) is in mm/rev or in/rev

Feed rate selection (special) is in m/min or in/min

Rapid traverse rate different for X and Z axes

Multiple repetitive cycles for turning, boring, facing, contour repeat, grooving, and threading are available

Feed rate override is common from 0 to 200% in 10% increments (on some lathes only from 0 to 150%)

X-axis can be mirrored

Tailstock can be programmable

Automatic chamfering and corner rounding using C and R (or I / K) addresses in G01 mode

Thread cutting feed rate available with six-decimal place accuracy (for inch units)

Thread cutting feed rate available with five-decimal place accuracy (for metric units) - seldom needed

Least input increment in X-axis is 0.001 mm or 0.0001 inches on diameter - one half of that

5. PROGRAM PLANNING

Development of any CNC program should always begin with a very carefully planned process. Such process typically starts with an engineering *drawing* (also called a *blueprint* or a *technical print*) of the required part, released for production. Before any machining process can be completed, several steps must be considered and carefully evaluated. Placing a greater effort into program planning will yield better program and better machined part.

A. STEPS IN PROGRAM PLANNING

Individual steps required in program planning are generally determined by the nature of part to be machined. There is no magic overall formula available for all jobs, but some basic steps are quite common and should always be considered carefully:

- Initial information provided / Machine tools features
- Part complexity / Evaluation of machining features
- Manual programming / Computerized programming
- Typical programming procedure / Program structure
- Part drawing / Engineering data
- Methods sheet / Material specifications
- Machining sequence - Operations / Tool order
- Tooling selection / Cutting Holders / Inserts / HSS Tools
- Part setup / Part holding / Fixtures
- Technological decisions / Cutting conditions
- Work sketch and individual calculations
- Quality considerations in CNC programming

All steps in the list are suggestions only - they are *guidelines*. Individual steps should always be flexible, so they can be adapted for any job and its unique requirements.

B. INITIAL INFORMATION

The main purpose of most engineering drawings is to define the part shape, individual dimensions, and relationships between part features. Some drawings may also include data about the initial blank material (stock), such as type, size, and shape. In CNC programming, good familiarity with various materials is important. For programming purposes, materials used to machine a part are evaluated by their size, type, shape, condition, hardness, etc.

Part drawing and material data are the primary sources of information about a specific part to be machined. They define the starting point of program planning. The objective of such a plan is to collect all available data and use all initial information for one purpose - to establish grounds for the most efficient method of machining, along with all related considerations - mainly part accuracy, productivity, safety, and convenience.

Drawing and material data provide much of initial information, but they are not the only source. A great part of what is needed to develop a part program is not found in the drawing directly, but in other documentation. For example, a process sheet (routing sheet) provides many engineering requirements *not covered* in the drawing, such as *pre-* and *post-* machining operations, grinding allowances, assembly features, requirements for hardening, next machine setup, and many others. Collecting relevant information from all available sources provides enough groundwork to start planning a CNC program development.

C. CNC MACHINE FEATURES

No amount of initial information is much useful if the selected CNC machine is not *suitable* for a particular job. During program planning, CNC programmer concentrates on a *particular machine tool with a particular CNC system*. These two major parts of a CNC machine are always connected, and they must always be considered in any single CNC machine definition. It is just not enough to select a special fixture or a special setup - the CNC machine itself has to be suitable to handle any required setup.

Modern technology offers a large number of special features that can be purchased as options for the selected CNC machine. These options are too numerous to list, but any manufacturer's or dealer's web sites specify all details. When a CNC machine is purchased and delivered, the machine shop needs and requirements should be satisfied, at least for a few years. Very few companies go and buy a new CNC machine just to suit a particular job, although that is not an uncommon situation. Yet, such cases are rather rare and happen only if they make strong economic sense.

Machine Type and Size

Possibly the two most important steps in program planning relate to the *type* and *size* of CNC machine, particularly its *workspace* or *work area*. Other features, equally important, are machine power rating, spindle speed and

feed rate range, number of tool stations, tool changing system, available accessories, etc. Typically, small CNC machines have higher spindle speeds and lower power rating, large machines have lower spindle speeds available, but their power rating is higher.

Control System

Control system is the heart of a CNC machine. Being familiar with all standard and optional control features available is a must. This knowledge enables using a variety of advanced programming methods, such as different machining cycles, subprograms, macros and other time saving features of a modern CNC system.

A programmer does not have to physically run a CNC machine. Yet, part programs will become better and more creative with good understanding of the machine and its control system. Program development reflects programmer's knowledge of various CNC machine operations.

One of the main concerns in program planning should be how the CNC operator *perceives* the part program. To a large degree, such perception is quite subjective, at least in the sense that different operators will express their personal preferences, often in opposite terms. On the other hand, every operator appreciates an error-free, concise, well documented and professionally prepared part program, written in a consistent way, one after another. A poorly designed program is disliked by any operator, regardless of personal preferences.

D. PART COMPLEXITY

When all initial information, such engineering drawing, material stock, available tools and CNC equipment are evaluated, the complexity of programming task becomes much clearer. How difficult is to program a given part manually? What are the machine capabilities? What are the costs? Many questions have to be answered before starting actual program development.

Many simple programming jobs may be assigned to a less experienced programmer or even to a CNC operator. It makes sense from the management perspective, and it is also a good way to gain valuable experience.

Difficult or complex jobs will benefit from a computerized programming system. Technologies such as *Computer Aided Design* (CAD) and *Computer Aided Manufacturing* (CAM) have been a strong part of the manufacturing process for many years. The cost of a CAD/CAM system is only a fraction of what it used to be only a few years ago. Even small shops now find that benefits

offered by modern technology are too significant to be ignored. Several programming systems are available and can handle virtually any job. For a typical machine shop, a Windows based programming software can be very beneficial. Atypical example of this kind of application is the very popular and powerful *Mastercam™* or *Edgecam*. There are several others. All major software vendors offer three-axis milling, standard lathe, multi-axis milling, turning with milling axes (live tooling), five axis machining, wire EDM, etc.

E. MANUAL PROGRAMMING

Manual programming (without a computer) has been the most common method of preparing a part program for many years. The latest CNC controls make manual programming much easier than ever before by using fixed or repetitive machining cycles, variable type programming, graphic tool motion simulation, standard mathematical input and other time saving features. In manual programming, all calculations are done by hand, with the aid of a pocket calculator - no computer programming is used. Programmed data can be transferred to CNC machines using a *memory card* or via a *cable*, using an inexpensive desktop or a laptop computer. Either process is faster and more reliable than other methods. Short programs can also be entered manually, by keyboard entry, directly at the machine. A punched tape used to be common media of the past but has virtually disappeared from machine shops.

Disadvantages

There are some disadvantages associated with manual programming. Perhaps the most common is the length of time required to actually develop a fully functioning CNC program. Manual calculations, verifications and other related activities in manual programming are very time consuming. Other disadvantages, also very high on the list, are a large percentage of errors, the difficulty in making changes to a program, a lack of tool path verification (simulators are available, such as *NC Plot*), and many others.

F. CAD/CAM AND CNC

The need for improved efficiency and accuracy in CNC programming has been the major reason for development of a variety of methods that use a computer to prepare part programs. Computer assisted CNC programming has been around for many years. First, in the form of language-based programming, such as *APT™* or *Compact II™*. Since the late 1970's, CAD/CAM has played a significant role by adding visual aspect to the programming process. The acronym *CAD/CAM* means *Computer Aided Design* and *Computer Aided Manufacturing*. The first three letters *CAD* - cover the area of engineering design and drafting. The second three letters - *CAM* - cover the area of computerized manufacturing, where CNC programming is only a small part. The whole subject of CAD/CAM covers much more than just design, drafting and programming. It is a part of modern technology also known under yet another acronym - *CIM* - *Computer Integrated Manufacturing*.

In the area of numerical control, computers have played a major role for a long time. Machine controls have become more sophisticated, incorporating the latest techniques of data processing, storage, tool path graphics, machining cycles, etc. Programs can now be prepared with the use of inexpensive computers, including sophisticated graphical interface. Initial cost should no longer be an issue, even small machine shops are able to afford a programming system in house. These systems are also popular because of their flexibility. Typical computer-based CNC programming system does not have to be dedicated only to programming - all related tasks, often done by the programmer, can also be implemented on the same computer. For example, cutting tool inventory management, database of part programs, material information sheets, setup sheets and tooling sheets, etc. The same computer could also be used for uploading and downloading CNC programs.

Integration

The key letter in acronym *CIM* is - *integration*. It means putting all elements of manufacturing together and work with them as a single unit and more efficiently. The main idea behind successful

integration is to avoid duplication. One of the most important rules of using a CAD/CAM computer software (and CNC) is:

NEVER DO ANYTHING TWICE!

When a drawing is developed in CAD software (such as *AutoCAD*), then done again in CAM software (such as *Mastercam*), there is duplication. Duplication breeds errors. In order to avoid duplication, most of CAD systems incorporate a transfer method of the design to a selected CAM system to be used for CNC programming. Typical transfers are achieved through special DXF or IGES files. The *DXF* stands for *Data Exchange Files* or *Drawing Exchange Files*, and the *IGES* abbreviation is a short form of *Initial Graphics Exchange Specification* files. Once the geometry is transferred from CAD system to CAM system, only the tool path related process is added. Using a *post processor* (special kind of formatter), the computer software will prepare part program, ready to be loaded directly to the CNC machine.

In most cases, both DXF and IGES methods have been replaced by direct translators, when CAD/CAM vendors provide a facility in their software to import a native file, without converting it first.

Future of Manual Programming

It may seem that manual programming is on the decline. In terms of actual use, this is quite likely true. However, it is necessary to keep in perspective that any computerized technology is based on the already well-established methods of manual programming. Manual programming for CNC machines serves as the *source* of new technology - it is the very elementary concept on which computerized programming is based. This knowledge base opens doors for development of more powerful hardware and software applications.

Manual programming may be used somewhat less frequently today and eventually will be used even less - but knowing it well - *really understanding it* - is and always will be the key to harness the power of CAM software. Even computers cannot do everything. There will always be some special programming project that no CAM software, regardless of price, may handle to absolute satisfaction. If machine control system can handle such a project, manual programming is the way to have ultimate control, when any other methods may not be suitable or practical.

Even with a well customized and organized computerized programming system, how can the generated program output be exactly as intended? How can a CNC operator change any program entry at the machine, without knowing its rules and structure?

Successful use of computerized programming requires understanding of manual programming methods.

G. TYPICAL PROGRAMMING PROCEDURE

Planning a CNC program is not much different than any other planning - at home, at work, or elsewhere - it must be approached in a *logical and methodical* way. The first decisions relate to *what tasks* have to be done and *what goals* have to be achieved. All other decisions relate to *how* to accomplish these goals in an efficient and safe manner. Such a progressive method of approach not only isolates individual problems as they develop, but it also forces their solution *before* any next step can be taken.

The list of following items forms a fairly common and logical sequence of tasks to be done in CNC programming. All items are listed in a *suggested order* only, offered for further and deeper evaluation. This - or any other - order should be changed to reflect special conditions or working habits. Some items may be missing from the list altogether, others may be somewhat redundant:

Study of initial information (drawing and methods)

Material stock (blank) evaluation

- Machine tool specifications
- Control system features
- Sequence of machining operations
- Tooling selection and arrangement of cutting tools
- Part setup on the machine
- Technological data (speeds, feed rates, etc.)
- Tool path determination
- Working sketches and mathematical calculations
- Program writing and preparation for transfer to CNC
- Program testing and debugging
- Program documentation

There is only one goal in CNC program planning - to provide all instructions in the form of a program that will result in an error-free, safe and efficient CNC machining. Individual steps in the suggested procedures may require some changes - for example, should cutting tools be selected *before* or *after* the final part setup is determined? Can manual part programming method be used efficiently? Are any working sketches necessary? Do not be afraid to modify any so-called *ideal* procedure - either temporarily, for a given job, or even permanently, in order to reflect a particular CNC programming style.

Remember, there are no ideal procedures

H. PART DRAWING

Part drawing is the single most important document used in CNC programming. It visually identifies the shape, important dimensions, tolerances, surface finish and many other requirements for the completed item. Drawings of complex parts often require many sheets, showing different views, details and sections. CNC programmer first evaluates *all* drawing data first, then isolates those that are relevant for a particular program development. Unfortunately, many drafting methods do not reflect the actual CNC manufacturing process. They reflect the *designer's thinking*, rather than the method of manufacturing. Such drawings are generally correct in technical sense, but they are harder to study by the programmer and may need to be '*interpreted*' to be of any value in CNC programming. Typical examples are methods of applying dimensions, absence of a datum point that can be used as a program reference point and the view orientation in which the part is drawn. In the CAD/CAM environment, the traditional gap between the design, drafting and CNC programming must be eliminated. Just as it helps the programmer to understand designer's intentions, it helps the designer to understand the basics of CNC programming. Both, the designer and the programmer have to understand each other's methods and find common ground that makes the whole process of design and manufacturing coherent and efficient.

Title Block

Drawing title block - *Figure 6-1* - is typical to all professional drawings. Its purpose is to collect all descriptive information related to a particular drawing.

No.	Date	Revision	By
Part Name:			
Scale:		Material:	
Dr.:		Date:	
Chk.:		Drawing number:	
App.:			

Figure 6-1

A title block example of an engineering drawing

The size and contents of a title block vary between companies, depending on the type of manufacturing and internal standards. It is usually a rectangular box, positioned in the corner of a drawing, divided into several small boxes. Contents of the title block include such items as the part name and part number, drawing number, material data, revisions, special instructions, etc. Data in the title block supply crucial information for CNC programming and can be

used for program documentation to make easier cross referencing. Not all title block information is needed in programming but may be used for program documentation.

Revision dates in a drawing are associated with the title block. They are important to the programmer, as they indicate *how current* is the drawing version. Only the latest version of a part design is important to manufacturing.

Dimensioning

Part drawing dimensions are either in *metric* or *imperial* units. Individual dimensions can be referenced from a certain datum point, or they can be consecutive, measured from the previous dimension. Often, both types of dimensions are mixed in the same drawing. When writing a program, it may be more convenient to translate all consecutive - or *incremental* - dimensions into datum - or *absolute* - dimensions. Most CNC programs benefit from drawings using *datum*, or *absolute dimensioning*. Similarly, when developing a subprogram for tool path translation, an *incremental* method of programming may be the right choice - and any choice depends on the application. The most common programming method for all CNC machines uses *absolute* dimensioning method - Figure 6-2, mainly because of editing ease within the CNC system (at the machine).

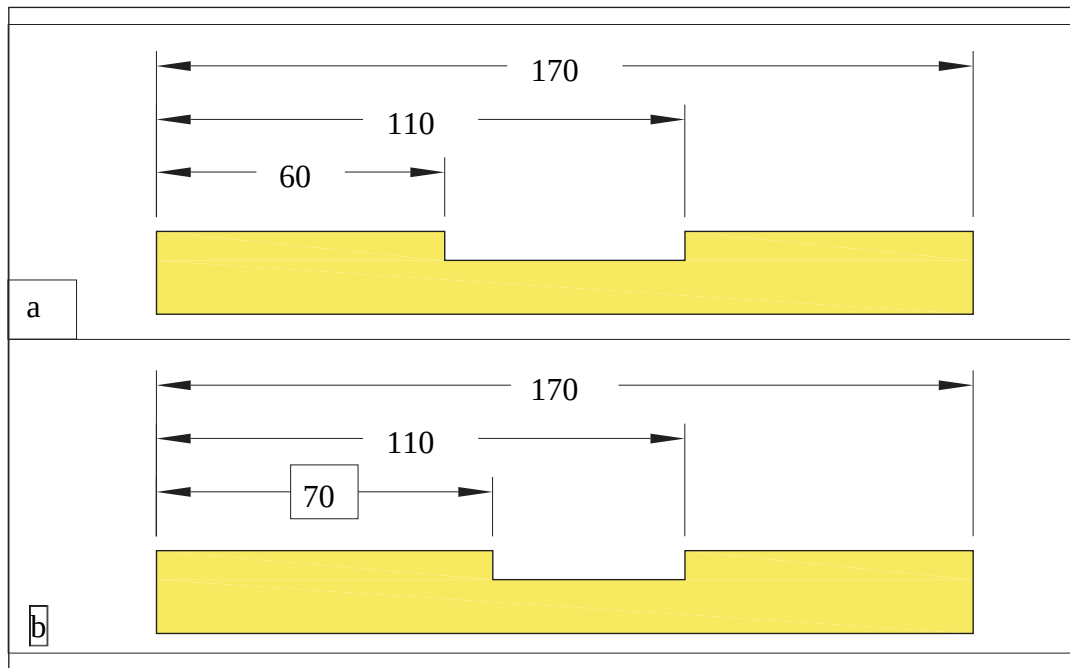


Figure 6-2

Program using *ABSOLUTE* dimensions

Only a single change in the program is necessary

With the absolute system of dimensioning, many program changes can be done by a single modification. Incremental method requires at least two modifications. The differences between the two dimensioning systems can be compared in *Figure 6-2*, using the *absolute* dimensioning method, and in *Figure 6-3*, using the *incremental* dimensioning method. The word *incremental* is more common in CNC, in drafting the equivalent word would be *relative*. Both illustrations show the *a)* figure before revision, and the *b)* figure after revision.

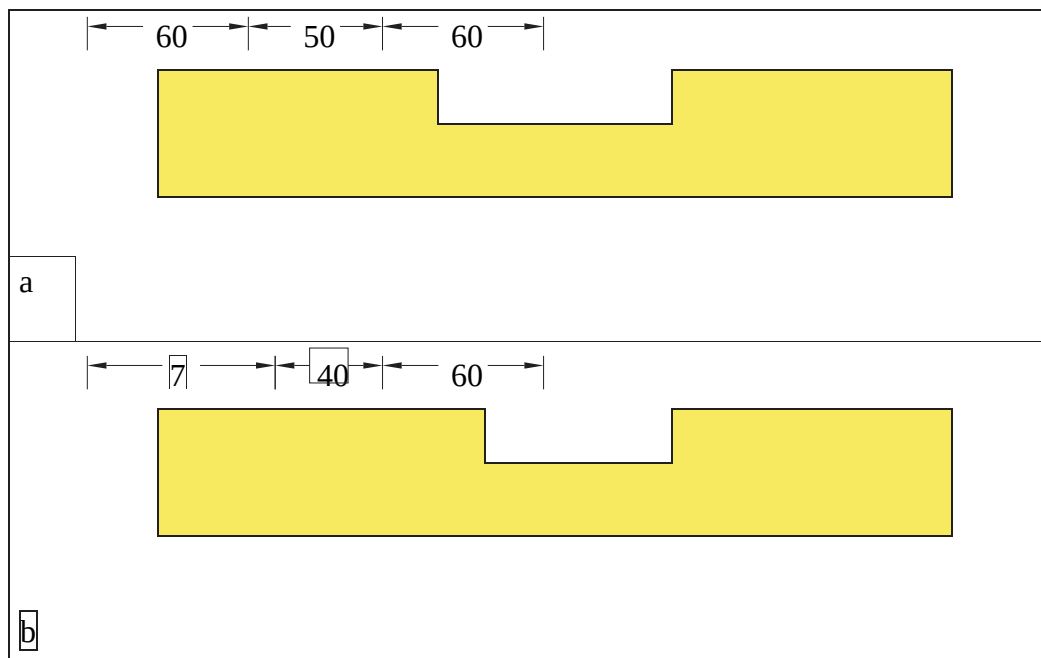


Figure 6-3

Program using *INCREMENTAL* dimensions

Two or more changes in the program are necessary

Fractions

Drawings (particularly older drawings) dimensioned in imperial units often contain fractions. A fractional dimension was sometimes used to identify a *less important* dimensional tolerances (such as ± 0.03 inches from the nominal size). The number of digits following decimal point often indicated a tolerance (the more digits specified, the smaller the tolerance range). These methods of using *implied tolerances* are not an ISO standard and are of no use in CNC programming. Fractional dimensions have to be changed into their decimal equivalents. The number of decimal places in a program is determined by the minimum increment of the control system. A dimension of 3-3/4 is programmed as 3.75 without rounding, and a dimension of 5-11/64 inches is programmed as 5.1719, its closest rounding. Many companies have upgraded their design standards to the ISO system and adhere to the principles of CNC dimensioning. In this respect, drawings using metric units are much more practical.

Some dimensioning problems are related to an improper use of CAD software, such as AutoCAD. Some designers do not change default settings for the number of decimal places and every metric dimension ends up with three decimal places and every imperial dimension with four decimal places. This is a poor practice and should always be avoided. The most professional approach is to specify dimensional tolerances for all dimensions that require them, and even use *Geometric Dimensioning and Tolerancing* standards (GDT or GD&T). GD&T establishes relationship between two or more features of a drawing. For example, GD&T dimension will define concentricity of a hole with a surface. Before GD&T became part of design drafting, written messages and instructions cluttered the drawing. These drawings do not conform to international ISO standards.

Degrees

Angular dimensions should always be specified in decimal degrees (DD), according to modern standards. Older drawings will often contain angular dimensions specified in *degrees-minutes-seconds* (DMS or D-M-S). To convert

DMS to DD:

DD	D	M	_____	S
60	3600			

+ where...

DD	=	Decimal degrees	
D	=	Degrees	(as per drawing)
M	=	Minutes	(as per drawing)
S	=	Seconds	(as per drawing)

Tolerances

For high quality precision machining work, many part dimensions are specified within a certain *range of acceptable deviation from their nominal size*. For example, imperial units' tolerance of $+0.001/0.000$ inches will be different from metric tolerance of $+0.1/0.0$ mm. Dimensions of this type are usually critical dimensions and require special attention during CNC machining. It may be true in many companies that CNC operators are ultimately responsible for maintaining required part sizes within tolerances (providing the program is correct) - but it is equally true, that CNC programmers can make machine operator's task a bit easier. Consider the following lathe example:

Drawing dimension specifies a certain size as $75+0.00/-0.05$ mm. *What actual dimension should appear in the program?*

Programmer has several choices here. High side of the dimensional range may be programmed

as X75.0, low side as X74.95. A middle value of X74.975 is also an option. Of course, any other dimension between high and low value is also a possibility, although not a practical one. Each selection is mathematically correct. A creative CNC programmer looks not only for mathematical aspects, but for technical aspects as well. Cutting edge of a tool wears out with more parts machined. That means the machine operator has to *fine-tune* part sizes by using *tool wear offsets*, available on most CNC systems. Such a manual interference during machining process is acceptable, but when done too often, it slows down the production and adds to overall costs.

Properly focused programming approach can control frequency of such manual adjustments to a great degree. Consider the 75 mm mentioned earlier. If it is an *external* diameter, the tool edge wear will cause the actual dimension during machining to become progressively *larger*. In case of an *internal* diameter, the actual dimension will become *smaller* as the cutting edge wears out. By programming X74.95 for the external diameter (bottom limit) or X75.0 for the internal diameter (top limit), the cutting-edge wear will move *into* the tolerance range, rather than *away* from it. Manual tool offset adjustment at the machine may still be required, but less frequently. Another approach is to select *middle size* of the tolerance range - this method will also have a positive effect, but more manual adjustments may be necessary during machining.

Surface Finish

Precision machined parts always require a certain degree of surface finish quality. Related drawings indicate desired surface finish quality for various part features. Imperial drawings define surface finish quality in *micro inches*, where $1 \text{ micro inch} = 0.000001''$. Metric drawings use surface requirements expressed in *microns*, where $1 \text{ micron} =$

0.001 mm . Symbol for a micron is the Greek letter. Some drawings use standard symbols - Figure 6-4.

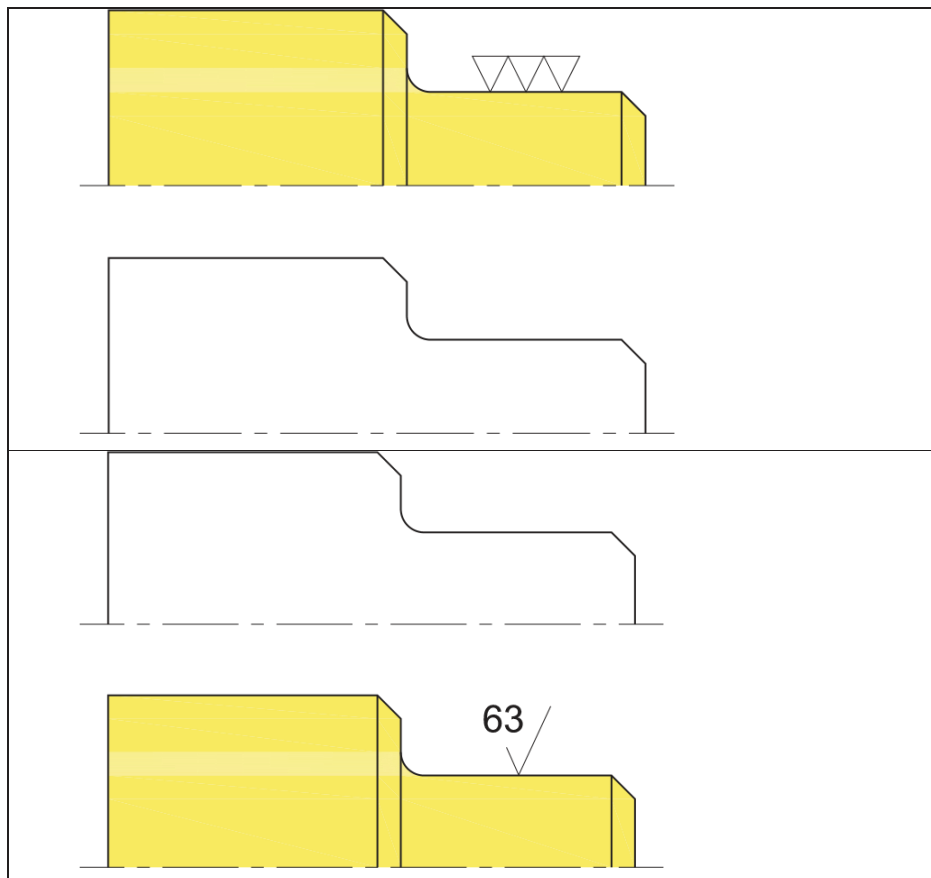


Figure 6-4

Surface finish marks in a drawing:
Metric (top) and imperial (bottom)

The most important factors influencing quality of surface finish are spindle speed, feed rate, cutting tool radius and the amount of material removed. Generally, a larger cutter radius and slower feed rates contribute towards *finer* surface finishes, while the opposite is equally true. Overall cycle time will be longer but can often be compensated for by elimination of any subsequent operations such as grinding, honing or even lapping.

Drawing Revisions

Another important section of part drawings, often overlooked by CNC programmers, shows engineering changes (known as *revisions*) made on the drawing up to a certain date. Using reference numbers or letters (*Figure 6-5*), the designer identifies such changes, usually with both values the previous and the new value - for example:

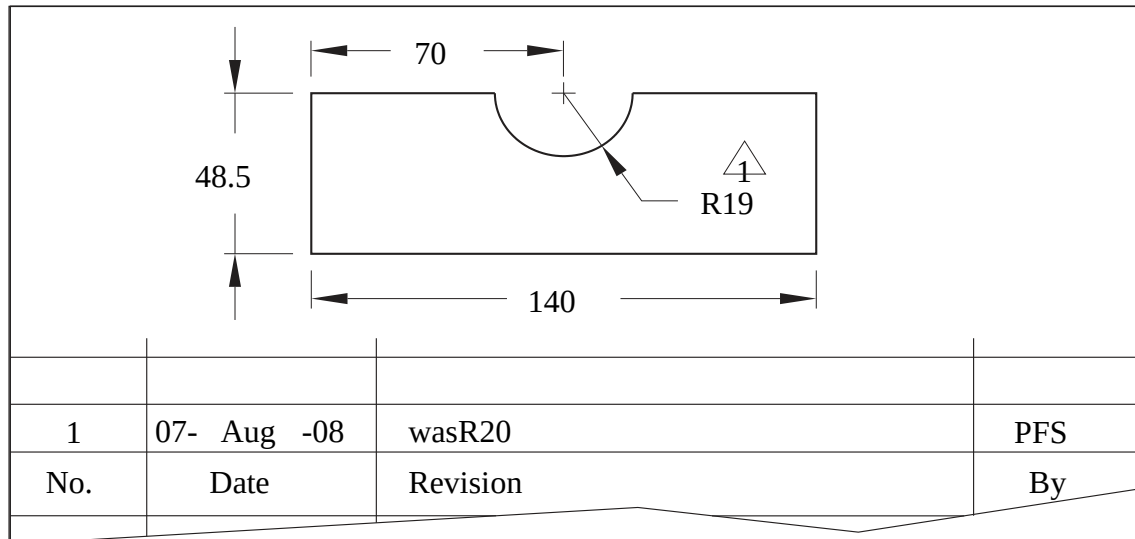


Figure6-5

Drawing revision example

Only the *latest* changes for each dimension are important to CNC program development. Make sure such a program not only reflects the current (latest) engineering design, but also is identified in some unique way to distinguish it from any previous program versions. Many programmers keep a copy of the part drawing corresponding to the stored program, thus preventing possible misunderstandings later.

Special Instructions

Many drawings also include special instructions and comments that cannot be expressed with established drafting symbols and are therefore spelled out independently, in words. Such instructions are very important for CNC program planning, as they may significantly influence the programming procedure. For example, a certain part element is identified as a *ground* surface or diameter. Drawing dimensions always show the *finished* size. During program development, this dimension likely has to be adjusted for any grinding allowance necessary. Actual amount of such allowance is selected by the programmer and should be written as a special instruction in the program. Another example of special instructions relates to any machining done during part assembly. For example, a certain hole should be drilled and tapped, and is dimensioned exactly as any other hole, but a special instruction indicates the drilling and tapping must be done when the part is handled during assembly. Operations relating to such a hole are *not* programmed and any overlook of small instruction such as this example, may result in an unusable part.

Many drawing instructions use a special pointer called a *leader*. Usually, it is a line ending with an arrow, pointing towards the specific area that it relates to. For example, a leader may be pointing to a hole, with a worded caption:

12 REAM 2 HOLES - 20 DP

This is a requirement to ream 2 holes to the full depth of 20 mm with a reamer that has 12 mm diameter.

I. METHODS SHEET

Some companies have a staff of qualified manufacturing technologists or process planners responsible for determination of all manufacturing processes. Their responsibility is to develop a series of machining instructions, detailing the exact route of each part through the manufacturing process. They allocate work to individual machines, develop machining sequences and setup methods, select tooling, etc. Their instructions are written in a *methods sheet* (also called a *routing sheet*) that accompanies the part through all stages of manufacturing, typically in a plastic folder (or as a computer file). If such a sheet is available, its copy should become part of CNC documentation. One main purpose of a methods sheet is to provide CNC programmer with as much information as possible to shorten turnaround between programs. One great advantages of methods sheets in programming are their comprehensive coverage of all required operations, both CNC and conventional, thus offering a complete overview of the manufacturing process. High quality methods sheet will save a lot of decisions - if it is made by a qualified manufacturing engineer, who specializes in work detailing. The ideal methods sheet is one where all recommended CNC manufacturing processes closely match established part programming methods.

For various reasons, many CNC machines shops do not use methods sheets, routing sheets or similar documentation. CNC programmer acts as a process planner as well. Such an environment offers a certain amount of flexibility but demands a large degree of knowledge, skills and responsibility from one person at the same time.

J. MATERIAL SPECIFICATIONS

Also, important consideration in program planning is evaluation of the *material stock*. Typical material is raw and unmachined (a bar, billet, plate, forging, casting, etc.). Some material may be already pre-machined, routed from another machine or operation. It may be solid or hollow, with a small or a large amount to be removed by CNC machining. The *size* and *shape* of material determines the setup mounting method. The *type* of material (steel, cast iron, brass, etc.) will influence not only the selection of cutting tools, but the cutting conditions for machining as well.

A program cannot be planned without knowing the type, size, shape and condition of the material

Material Uniformity

Another important consideration, often neglected by programmers and managers alike, is the *uniformity* of material specifications within a particular batch or from one batch to another. For example, a casting or forging ordered from two suppliers to be used for the *same part* may have slightly different sizes, hardness and even shape. Another example is a material cut into single pieces on a saw, where the length of each piece varies beyond an acceptable range. This inconsistency between blank parts makes programming more difficult and time consuming. It also creates potentially unsafe machining conditions. If such problems are encountered, the best planning approach is to place emphasis on machining safety than on machining time. At worst, there will be some *air cutting* or slower than needed cutting feed rate, but no machining cuts will be too heavy for the tool to handle.

Yet another approach is to separate non-uniform material into groups and make separate programs for each group, properly identified. The best method is to cover all known and predictable inconsistencies under program control, for example, using the block skip function.

Machinability Rating

Also, very important aspect of material specification is its *machinability*. Charts with suggested speeds and feeds for most common materials are available from major tooling companies. These

charts are helpful in programming, particularly when an unknown material is used. The suggested values are a good starting point, and can be optimized later, when the material properties are better known. See *page 523* for some very basic and general suggestions.

Machinability rating in the imperial units is given in units called *feet per minute* (ft/min). Often the terms *surface feet per minute*, *constant surface speed* (CSS), *cutting speed* (CS), *peripheral speed* or just *surface speed* are used instead. For metric designation of machinability rating, *meters per minute* (m/min) are used. In both cases, the *spindle speed* (r/min) for a given tool diameter (for a mill) or a given part diameter (for a lathe) is calculated, using common formulas. For imperial units, the spindle speed can be calculated in *revolutions per minute* (r/min):

$$\text{r / min} = \frac{12 \text{ ft / min}}{D}$$

For metric calculations, the formula is similar:

$$\text{r / min} = \frac{1000 \text{ m / min}}{D}$$

+ where ... r/min = Revolutions per minute (spindle speed S)

12 = Converts feet to inches

1000 = Converts meters to millimeters

ft/min = Cutting speed in feet per minute

m/min = Cutting speed in meters per minute

(pi) = Constant value of 3.141593...

D = Tool diameter (milling) or

MACHI **NI** Part diameter (turning) - in inches or mm

NG SEQUENCE

Machining sequence defines the order of machining operations. Technical skills and machine shop experience do help in program planning, but a good quantity of common-sense approach is equally important. Machining sequences must have logical order - for example, drilling must be programmed before tapping, roughing operations before finishing, first operation before second, etc. Within this logical order, further specification of the order of individual tool motions is required for a particular tool. For example, in turning, a face cut may be programmed on the part first, then roughing all material on diameters will take place. Another method is to program a roughing pass for the first diameter, then face and continue with the remainder of the diameter roughing afterwards. In drilling, a spot drill before drilling may be useful for some applications, but in another program a center drill may be a better choice. There are no fixed rules, no absolutes, which determine which method is better - each CNC programming assignment has to be considered individually, based on the basic criteria of safety, quality, and efficiency.

Machining sequence must follow logical order

Basic approach for determining a machining sequence is careful evaluation of all related operations. In general, program should be planned in such a way that the cutting tool, once selected, will do as much work as possible, before a tool change. On most modern CNC machines, much less time is needed for positioning a tool than for a tool change. Another consideration is in benefits gained by programming all heavy operations first, then the lighter semi finishing or

finishing operations. It may mean an extra tool change or two, but this method minimizes any shift of material in the holding fixture while machining. Another important factor is the current position of cutting tool when a certain operation is completed. For example, when drilling a pattern of holes in the order of 1-2-3-4, the next tool (such as a boring bar, reamer or a tap) could be programmed in opposite order of 4-3-2-1 to minimize unnecessary tool motions - *Figure 6-6*.

T01 = Spot ill	T02 = Drill	T03 = Tap
Hole 1	Hole 4	Hole 1
Hole 2	Hole 3	Hole 2
Hole 3	Hole 2	Hole 3
Hole 4	Hole 1	Hole 4

Figure 6-6

Typical machining sequence for three common hole operations

This machining sequence may have to be changed after final selection of tools and setup method. Although practical in many cases, the reverse machining sequence may not be practical when stored as subprograms. This important subject starts on *page 383*).

Program planning is not an independent execution of individual steps - it is a very interdependent and very logically and coherent approach to achieve a certain goal.

K. TOOLING SELECTION

Selecting tool holders and cutting tools is another important step in planning a CNC program. Category of tooling, their selection and usage, includes a lot more than just cutting tools and tool holders - it covers an extensive line of accessories, involving numerous vises, fixtures, chucks, sub-tables, steady-rests, tailstock, indexing tables, clamps, collets and many other holding and auxiliary devices. Cutting tools require special attention, due to the very large variety available and their direct effect in machining.

Cutting tool used for the job is usually the most important selection. It should be selected by two main criteria:

Efficiency of usage

Safety in operation

Many supervisors responsible for CNC programming try to make the existing tooling work at all times. Often, they ignore the fact that a suitable new tool may do the job faster and more economically. A thorough knowledge of tooling and its applications is a separate technical profession - the programmer should know well all general principles of cutting tool applications. In many cases, a tooling company representative may provide additional valuable assistance.

Arrangement of tools in the order of usage is also a subject of serious consideration in CNC program planning. On CNC lathes, each cutting tool is assigned to a specific turret station, which requires consideration about the *distribution of tools* - how they are arranged between short and long tools (such as short turning tools versus long boring tools). This is important for prevention of a possible interference during cutting or tool changing. Another concern should be the *order* in which each cutting tool is called, particularly for machines that do not have a bi-directional tool indexing. Most machining centers use a random type of tool selection, where the order of tools is unimportant, only the diameter of the tool and its weight has to be considered.

All tool offset numbers and other program entries should be documented in a form known as the *tooling sheet*. Such a document serves as a guide to the operator during job setup. It should

include at least some basic documentation relating to the selected tool. For example, such documentation may include the tool description, its length and diameter, number of flutes, tool and offset numbers, speed and feeds selected for that tool and other relevant information.

L. PART SETUP

Another decision in program planning relates to the actual part setup - *how to mount* the raw or pre-machined material, what *supporting* tools and other devices should be used, how many *operations* are required to complete as many machining sequences as possible, where to select a *program zero*, etc. Setup is necessary and it should be done efficiently.

Some types of machines are designed to make the setup time more productive. Multi spindle machining centers or lathes can handle two or more parts at the same time. Special features, such as tool offset settings, bar feeder for a lathe, an automatic pallet changer or dual setup on the table, also help. Other solutions can be added as well.

Setup Sheet

At this stage of program planning, once the setup is decided, making a *setup sheet* is a good idea, particularly for jobs with many tools and/or operations. A setup sheet can be a simple sketch, designed mostly for use at the machine, which shows part orientation when mounted in a holding device, tool offset numbers used by the program, datum points and, of course, all necessary identifications and descriptions. Other information in a setup sheet should relate to some unique requirements established during planning stages of the program (such as position of clamps, bored jaws dimensions, depth of clamping, limits of tool extension, etc.). Setup sheet and tooling sheet can be combined into a single source of information. Most programmers use their own various versions.

M. TECHNOLOGICAL DECISIONS

The next stage of CNC program planning involves selection of spindle speeds, cutting feed rates, depth of cut, coolant application, etc. All of the already considered factors will have their influence. For example, the available range of spindle speeds is fixed for any CNC machine, size of the cutter and material type will influence speeds and feeds, power rating of the machine tool will help determine what amount of material can be removed safely, etc. Other factors that influence program design include tool extensions, setup rigidity, cutting tool material and its condition. Not to be overlooked is the proper selection of cutting fluids and lubricants - they, too, are important for overall part quality.

Cutter Path

Main core of any CNC programming is the precise determination of cutter path - known as the *tool path* or *toolpath*. This process involves individual cutter movements in its relationship to the part.

In CNC programming, always look at the cutting tool as being moved around the work!!

THIS PRINCIPLE APPLIES TO ALL CNC MACHINE TOOLS

The key factor for understanding this principle is to always visualize the *tool* motion, *not* the machine motion. The most noticeable difference between programming a machining center as compared to a lathe program is the cutter rotation as compared to part rotation.

In both cases, the programmer always must think in terms of the *cutter moving around the part* - Figure 6-7.

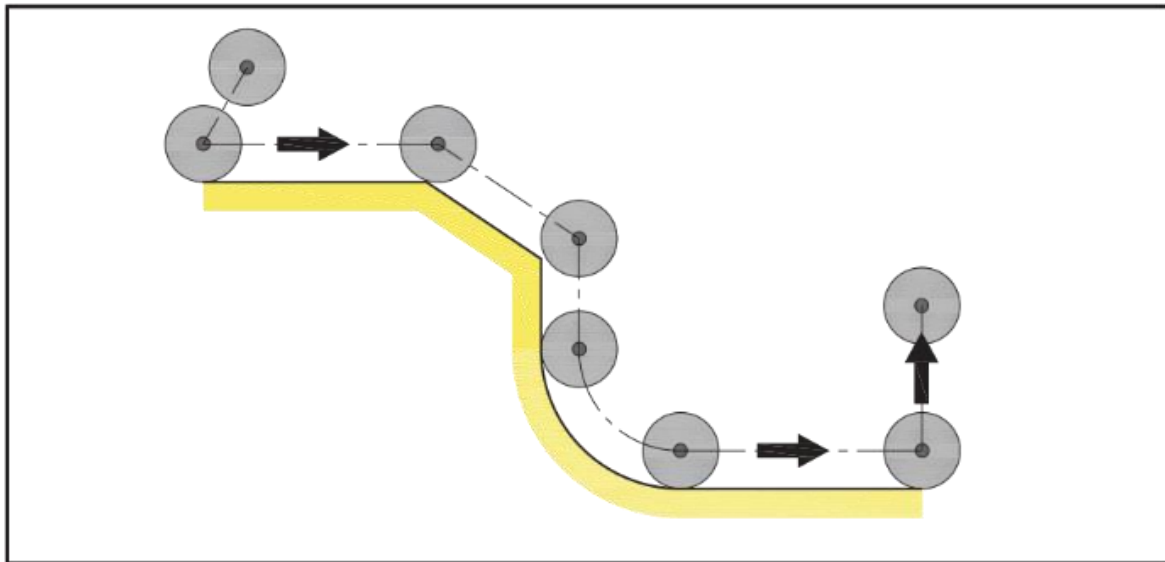


Figure 6-7

Contouring toolpath motion - as intended for milling or turning

Toolpath for all contouring tools always has to take into consideration the cutter *radius* (tool nose radius for turning and boring), either by programming an *equidistant toolpath* for the radius center or by using *cutter radius offset* (also known as *cutter radius compensation* or *tool nose radius compensation*). CNC machines for milling and turning are provided with rapid motion, linear interpolation and circular interpolation, all as standard features. To generate more complex paths, such as a helical milling motion or program rotation, a special option has to be available in the control unit. Two groups of typical toolpaths are used:

Point-to-point *also called* **Positioning**

Continuous *also called* **Contouring**

Positioning is used for a point location operation, such as drilling, reaming, tapping and similar operations; *continuous path* generates a contour (profile). In either case, all programmed data refer to the cutter position when a certain motion is completed.

This position is called *tool target position* - Figure 6-8.

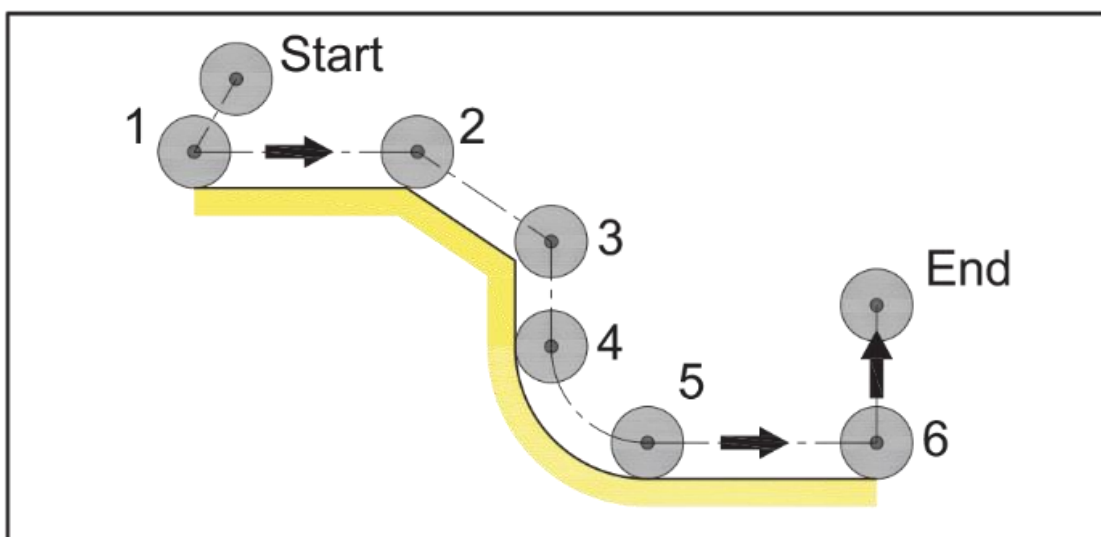


Figure 6-8

Contouring toolpath motion - target positions identified

The contour starts and end positions are identified and so are the positions for each contour

change. Each target position is called the *contour change point*, which has to be carefully calculated. The order of target locations in the program is very important. Based on the above illustration, it means that tool *position 1* is the *target* position commencing at the *Start* point, *position 2* is the *target* position beginning at point 1, *position 3* is the *target* from point 2 and so on, until the *End* position is reached. If the contour is for milling, these targets will be in X and Y axes. In turning, they will be in X and Z axes.

Most contouring operations require more than just one cutting motion, for example, roughing and finishing. Part of the programming process is to isolate areas that need roughing. Can one cutting tool do all operations? Can all tolerances be maintained? Does the tool wear a problem? Can the surface finish be achieved? When programming noncutting rapid motions, take the same care as with cutting motions. A particular focus should be to minimize rapid tool motions and ensure safe clearances.

Machine Power Rating

Machine tools are rated by their power, as one important specification for CNC programming. Laws of physics define that heavy cuts require more power than light cuts. A depth or width of a cut that is too large can break the tool and stall the machine. Such cases are unacceptable and must be prevented. CNC machine specifications list typically identifies power rating of the motor at machine spindle. This rating is either in *kW* (kilowatts) or *HP* (horsepower). Several formulas are available for calculations relating to various power ratings, establishing metal removal rates, tool wear factors, etc.

One of the more useful formulas is the comparison of *kW* and *HP* (based on 1 HP= 550 foot-pounds per second):

1 kW = 1.341022 HP	or	~ 1.3 HP
1 HP = 0.7456999 kW	or	~ 0.75 kW

The topic of power and forces in machining can be complex and is not always needed in everyday programming. There are publications devoted to this particular subject. Work experience is often a better teacher than formulas.

Coolants and Lubricants

When cutting tool contacts, the material for an extended period, great amount of heat is generated. Cutting edges get overheated, become dull and an insert or the whole tool may break. To prevent such possibilities, a suitable *coolant* must be used.

Water soluble, bio-degradable, oil is the most common coolant. A properly mixed coolant dissipates heat from the cutting edge, and it also acts as a lubricant. The main purpose of lubrication is to reduce friction and make the metal removal easier. Flood of the coolant should aim at the tool cutting edge, with a flexible pipe or through a coolant hole built into the tool itself.

Never use plain water as coolant it may severely damage the machine tool

CNC operator is responsible for a suitable coolant in the machine. Coolant should be clean and mixed in recommended proportions. Water soluble oils should be biodegradable to preserve the environment and properly disposed of. CNC programmer decides when to program coolant and when not. Ceramic cutting tools are normally programmed dry, *without a coolant*. Some cast irons do not require flood coolant, but air blast or oil mist may be allowed. These coolant functions vary between machines, so check the machine reference manual for further details.

Flood coolant may be used to cool down the part and gain better tolerances. It can also be used to flush away chips from congested areas, such as deep holes and cavities.

The main benefits of cutting fluids far outweigh their inconveniences. Cutting fluids are often messy, the cutting edge cannot be seen during operation, operator may get wet and sometimes old

coolant smells. With proper coolant management, all problems related to coolants can be controlled.

A coolant related programming issue is *when* to turn the coolant ON in the program. As the coolant ON function M08 only turns on the pump motor, make sure the coolant actually reaches the tool edge *before* contact with work. Programming coolant ON earlier is better than too late.

In summary, using coolant on CNC machining centers and lathes serves three major purposes:

- to dissipate heat from cutting edge and part
- to lubricate area between tool edge and part
- to flush away chips

N. WORK SKETCH AND CALCULATIONS

Manually prepared programs always require several mathematical calculations. This part of program preparation intimidates many programmers but is a necessary step, even for simple operations. Many complex contours will require more calculations, but not necessarily more complex calculations. Almost any mathematical problem in CNC programming can be solved by the use of arithmetic, algebra and trigonometry - that's it! Advanced fields of mathematics - analytic geometry, spherical trigonometry, calculus, surface calculations, etc. - are required for programming complex molds, dies and similar shapes. In such cases, a CAD/CAM programming system is necessary.

Those who can solve a right-angle triangle can make calculations for almost any CNC program. At the end of this digital book is an overview of some common mathematical problems and applications. When working with more difficult contours, it is often not the solution itself that is difficult, it is the ability to arrive at the solution. CNC programmer must have the ability to see exactly what triangle has to be solved, what formula to use, what approach to select. It is not unusual to do several intermediate calculations before the required final coordinate point can be established.

Calculations of any type often benefit from a visual (pictorial) representation. Such calculations usually need a working sketch. A simple sketch can be drawn by hand and should be done in an *approximate scale*. Larger sketch scales are much easier to work with. Making the sketch in at least approximate scale has one great advantage - *you can immediately see various relationships* - what dimensions should be smaller or larger than the others, the relationship of individual elements, the shape of an extremely small detail, etc. All that said, there is one purpose a sketch should *never* be used for, regardless how accurate it is:

Never use a scaled sketch to guess unknown dimensions!

Scaling a drawing is a poor and unprofessional practice, that creates more problems than it solves. It is a sign of either laziness or incompetence, commonly both.

Identification Methods

A sketch used for calculations can be done directly in the drawing, or on paper or even using CAD software. Every sketch is associated with several mathematical calculations - they make the sketch necessary in the first place. Using color coding or point numbering as identification methods offers benefits and better organization. Rather than writing coordinates at each contour change point in the drawing, use point reference numbers and create a separate coordinate sheet form using the reference numbers, as illustrated in *Figure 6-9*. A printable sample of a coordinate sheet is also included on the enclosed CD.

Position	X-axis	Y-axis	Z-axis

Figure 6-9

Coordinate sheet example - blank form (no data)

Such a sheet can be used for milling or turning, by filling only the applicable columns. The aim is to develop a consistent programming style from one program to another. Fill-in all values, even those that do not change.

A completed coordinate sheet is a better programming reference, as shown in Figure 6-10.

Position	X-axis	Y-axis	Z-axis
START	X57.0	Y126.0	
P1	X45.0	Y105.0	
P2	X108.258	Y105.0	
P3	X146.156	Y79.6	
P4	X146.156	Y50.0	
P5	X173.156	Y23.0	
P6	X238.0	Y23.0	
END	X238.0	Y62.0	

Figure 6-10

Coordinate sheet example - filled form for a milling toolpath

O. QUALITY IN CNC PROGRAMMING

All parts machined on CNC equipment are evaluated by their quality - usually *after* they are machined. Quality inspectors check many features - are the dimensions within tolerances? Is the surface finish up to standard? Is there a consistency between parts, etc.? Modern CNC equipment provides optional inspection related features during machining process, for example, *in-process gaging*. Many machine shops require their CNC machinists to be quality inspectors while the parts are being made. One subject that is equally important, yet not very often mentioned, is quality in CNC programming.

CNC programming starts with a plan. Although number one quality of a CNC programmer is knowledge and skill, there are at least two related and equally important considerations in program planning - programmer's *personal approach* and *professional attitude*. How the CNC programmer approaches a certain job, assignment or project will have a great influence on the final outcome of parts produced by the CNC operator. Programmer's attitudes have significant influence on program development and final results. They also have significant influence on the CNC operator - it's just human nature.

Ask yourself some questions. As a CNC programmer are you attentive to detail, are you precision minded, are you well organized, are you concerned if something is not done right? Do you 'cut corners' just to have the job done? Can a program you just developed, or an existing program, be improved further to make it safer and more efficient?

CNC program quality is much more than writing an error free program - that is the absolute prerequisite and goes without saying. Quality in programming includes concern as how a program effects the CNC operator, machine setup, and actual part machining. Quality in programming means constant effort at improvement and desire to make the next program even better.

Consistency in programming is one the best ways to achieve high quality programs. Once a certain method or process been found superior to others, stick with it. Use the same method again and again. CNC operators like nothing less than programs that vary in structure.

Part complexity should never stand in the way - it is only related to *your* knowledge level and willingness to solve problems. It should be a personal goal to make a program every program - the best program possible.

Set your quality standards high!

6. PART PROGRAM STRUCTURE

Typical CNC program is composed of series of sequential instructions related to machining a part. Each instruction is specified in a format the CNC system can accept, interpret and process. Each instruction must also conform to machine tool specifications. This program input method can be defined as *an arrangement of machining instructions and related tasks*, written in the format of a CNC system and aimed at a particular machine tool.

Various controls may have different formats, but most are similar. Subtle differences exist among CNC machines from different manufacturers, even those equipped with the same control system. This is common, considering the specific demands individual machine builders place upon the control manufacturer to accommodate many original and unique machine design features. Such variations are usually minor but still important for programming.

A. BASIC PROGRAMMING TERMS

The field of CNC has its own terminology, special terms and its jargon. It has its own abbreviations and expressions that only people in the field understand. CNC programming is only a small section of computerized machining, and it has a number of its own expressions. Most of them relate to the *structure* of part programs.

There are *four basic terms* used in CNC programming. They appear in professional articles, books, papers, lectures and so on. These words are the key to understanding general CNC terminology:

Character	Word	Block	Program

Each term is very common and equally important in CNC programming and deserves its own detailed explanation.

Character

A character is the smallest unit of CNC program. It can have one of three forms:

Digit

Letter

Symbol

Characters are combined into meaningful CNC *words*. This combination of digits, letters and symbols is called the *alpha-numerical* program input.

Digits

There are ten digits, 0 to 9, available for use in a program to create numbers. The digits are used in two modes - one for *integer* values (numbers without a decimal point), the other for *real numbers* (numbers with a decimal point). Numbers can have *positive* or *negative* values. On some controls, real numbers can be used with or without the decimal point. Numbers applied in either mode can only be entered within the range that is allowed by the control system.

Letters (Addresses)

All twenty-six letters of the English alphabet are available for programming, at least in theory. Most control systems will accept only certain letters and reject others. For example, a two-axis CNC lathe control will reject the letter Y, as the Y-axis is typically unique to milling operations (milling machines and machining centers). On the other hand, many CNC lathes with milling capabilities will accept the letter

(address) Y, if the Y-axis is available. Capital letters are normal designation in CNC programming, but some controls accept low case letters with the same meaning as their upper-case equivalent. ***If in doubt, use capital letters.***

Every control accepts CAPITAL letters but not all controls accept low case letters

Symbols

Several symbols are used for programming, in addition to the ten digits and twenty-six available letters. The most common symbols are the *decimal point*, *minus sign*, *percent sign*, *parenthesis* and others, depending on the control options. Their use in a program is strictly defined. Decimal point is used for values expressed in mm, inches or degrees. Minus sign is used to identify a dimensional value as negative, percent sign is used for file transfers, and parentheses are used for program comments and messages.

Word

A program *word* is a combination of *alpha-numerical* characters, creating a single instruction to the CNC. Each word begins with a capital letter, followed by a number representing a program code or actual value. Typical words indicate *axes position*, *feed rate*, *speed*, *preparatory commands*, *miscellaneous functions* and many others.

Word is the unit of instruction to the control system

Block

Just like one word is used as a *single* instruction to the CNC system, program block is used as a *multiple* instruction. A program entered into the control system consists of individual lines of instructions, sequenced in a logical order of processing. Each line - called a *sequence block* or simply a *block* - is composed of one or several *words* and each word is composed of *two or more characters*.

In the control system, each block must be separated from all others. To separate blocks in the MDI (*Manual Data Input*) mode at the control, each block has to end with a special *End-Of-Block* code (symbol). This code is marked as EOB on the control panel. When preparing part program on a computer using a keyboard, using the *Enter* key will terminate the block (similar to the old *Carriage Return* on typewriters). When writing a program on paper first, each program block should occupy only a single line on the paper. Each program block contains a series of individual instructions that are executed simultaneously.

Program

Part program structure may vary quite a bit for different controls, but the logical programming approach does not change from one control to another. CNC program usually begins with a *program number* or similar identification, followed by sequenced blocks of instructions in a logical order. The program ends with a *stop code* or a program termination symbol, such as the *percent sign* - %. Some controls also require the *stop code* at the program beginning. Internal documentation and messages to the CNC operator may be located in strategic places within the part program. Programming format has evolved significantly during the years and several formats have emerged.

B. PROGRAMMING FORMATS

Since the early days of numerical control, three formats had become significant in their time. They are listed in the order of their original introduction:

Tab Sequential Format no decimal point available ... old NC systems only

Fixed Format no decimal point available

... old NC systems only

Word Address Format decimal point available

... typically used with CNC systems

Only the *very early* control systems use the *tab sequential* or *fixed* formats. Both of them

disappeared in the early 1970's and are now obsolete. They have been replaced by much more convenient *Word Address Format*. Its greatest benefit is using addresses for words and decimal point format when necessary.

C. WORD ADDRESS FORMAT

The *word address format* is based on a combination of one letter and one or more digits - *Figure 7-1*.

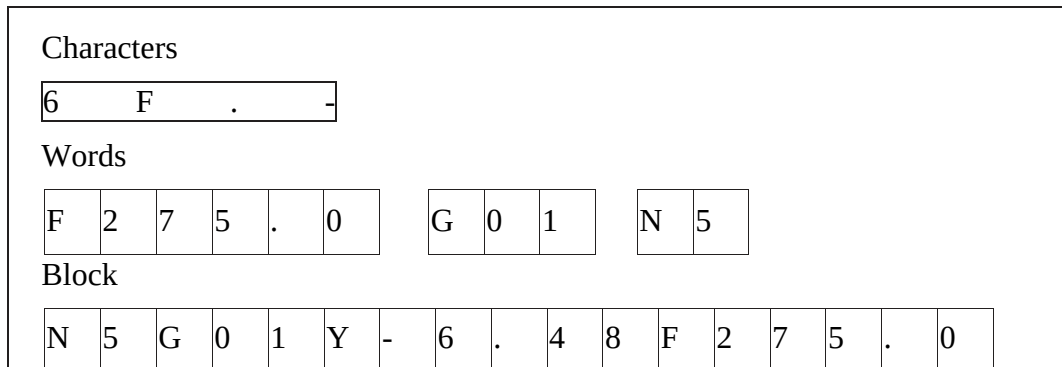


Figure 7-1

Typical word address programming format

In some applications, such a combination can be supplemented by a symbol, such as a minus sign or a decimal point. Each letter, digit or symbol represents one character in the program and in control memory. This unique alpha-numerical arrangement creates a *word*, where the letter is called the *word address*, followed by numerical data with or without symbols. The *word address* refers to a specific register of the control memory. Some typical words are:

G01 M30 D25 X15.75 N105 H01 Y0 S2500 Z-5.14 F12.0 T0505 T05 /M01 B180.0

The *address* (letter) in the block defines the *word meaning* and must always be written first. For example, X15.75 is correct, 15.75X is not. No spaces (space characters) are allowed *within* a word - X 15.75 is *not* correct - spaces are only allowed *before* the word, meaning before the letter, between words.

Data always indicate the word numerical assignment. This value varies greatly and depends on the preceding address. It may represent a sequence number N, a preparatory command G, a miscellaneous function M, an offset register number D or H, a coordinate word X, Y or Z, feed rate function F, spindle function S, tool function T, etc.

Any one *word* is a series of characters (at least two) that define a single instruction to the machine control unit. The above examples of typical words have the following meaning in a CNC program:

G01	<i>Preparatory command</i>
M30	<i>Miscellaneous function</i>
D25	<i>Offset number selection-milling applications</i>
X15.75	<i>Coordinate word-as positive value</i>
N105	<i>Sequence number (block number)</i>
H01	<i>Tool length offset number</i>
Y0	<i>Coordinate word-as zero value</i>
S2500	<i>Spindle speed function</i>
Z-5.14	<i>Coordinate word-as negative value</i>

F12.0	<i>Feed rate function</i>
T0505	<i>Tool function-turning applications</i>
T05	<i>Tool function-milling applications</i>
/M01	<i>Miscellaneous function/block skip symbol</i>
B180.0	<i>Indexing table function</i>

Individual program words are instructions grouped together to form logical sequences of programming code. Each sequence that will process one series of instructions simultaneously forms a unit called a *sequence block* or *program block* or - simply a *block*. The series of blocks is arranged in a logical order that is required to machine a *complete* part or a *complete* operation is called a *part program* also known as a *CNC program*.

The next block shows a rapid tool motion to the absolute position of X13.0Y4.6 within current units setting and with a coolant turned on:

N25 G90 G00 X13.0 Y4.6 M08 + where . . .

N25	Sequence or block number
G90	Absolute mode
G00	Rapid motion mode
X13.0 Y4.6	Coordinate location
M08	Coolant ON function

The control will always process any single block as one complete unit - *never partially*. Most controls allow a random word order in a block, as long as the block number is specified first. Many programmers follow an unofficial but recommended - order of various words in a block. For example, G-codes are listed first, followed by axes data, then all remaining instructions.

Block number must always be specified first

D. FORMAT NOTATION

Each program word can only be written in a specific way. The number of digits allowed in a word, depending on the address and maximum number of decimal places, is set by the control manufacturer. Not all letters can be used. Only letters with an assigned meaning can be programmed, except in a comment. Symbols can be used in only some words, and their position in the word is fixed. Some symbols are used only in custom macros - control limitations are important. Symbols supplement the *digits* and *letters* and provide them with an additional meaning. Typical programming symbols are the minus sign, decimal point, percent sign and a few others. All symbols are listed in a table on page 47.

Short Forms

Control manufacturers often specify the input format in an abbreviated form - *Figure 7-2*.

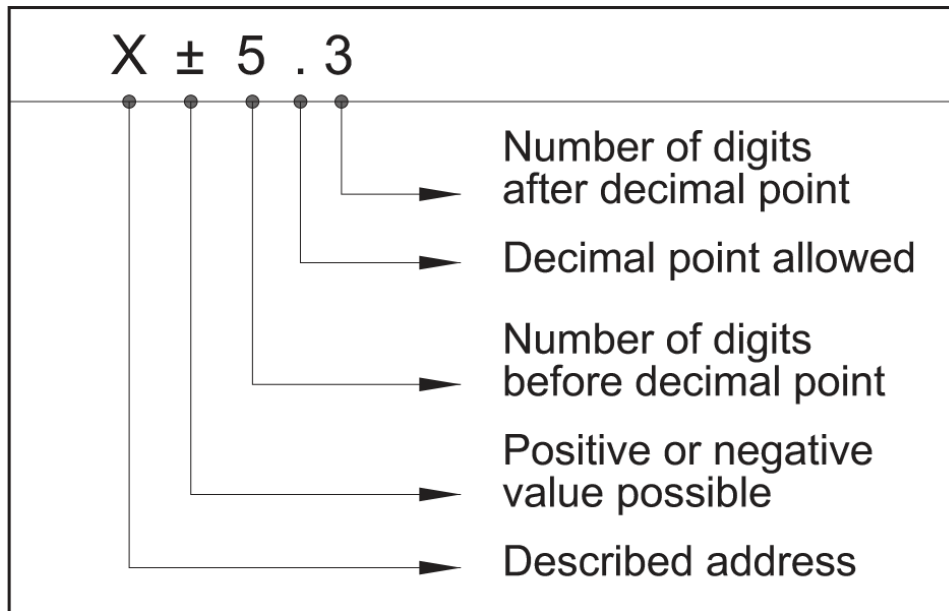


Figure 7-2

Word address format notation - X axis format in metric mode shown

Describing the full format description for each meaning would be unnecessarily too long. Consider the following complete and *not* abbreviated description of address X - as a coordinate word that is used in metric system:

Address X accepts positive or negative data with the maximum of five digits in front of a decimal point and three digits maximum after the decimal point - decimal point is allowed

Absence of decimal point in the notation means that decimal point is not used; the absence of a plus/minus ($\pm$) sign means that the address value cannot be negative - *a lack of sign means a positive value by default.*

These samples of format notation explain the shorthand:

G2 Two digits maximum, no decimal point or sign

N4 Four digits maximum, no decimal point or sign

S5 Five digits maximum, no decimal point or sign

F3.2 Five digits maximum, three digits maximum in front of the decimal point, two digits maximum behind the decimal point, decimal point is allowed, no sign is used

Be extra careful when evaluating shorthand notations from a control manual or other source. There are no industry standards and not all control manufacturers use the same methods, so the meaning of short forms listed may vary significantly. Typical list of word addresses, their format notation and description are shown in the following tables. They contain address notations based on typical Fanuc control systems.

Note:

The presented format notation concept is often applied by CAD/CAM software developers for CNC program output using post processors

Address	Notation	Description
Q	Q5.3 (Q4.4)	Depth of peck in fixed cycles G73 and G83
	Q5.3 (Q4.4)	Shift amount in fixed cycle G76 and G87
R	R5.3 (R4.4)	Retract point in fixed cycles Arc radius designation

S	S5	Spindle speed in r/min
T	T4	Tool function
X	X5.3 (X4.4)	axis coordinate value designation
	X5.3	Dwell function with G04 only
Y	Y5.3 (Y4.4)	Y axis coordinate value designation
Z	Z5.3 (Z4.4)	Z axis coordinate value designation
A	A5.3	Rotary or Indexing axis - unit is degrees - used about the X axis
B	B5.3	Rotary or Indexing axis - unit is degrees - used about the Y axis
D	D2	Cutter radius offset number (sometimes uses address H)
F	F5.3	Feed rate function - may vary
G	G2	Preparatory commands (G3 for extended G-code set)
H	H3	Offset number (tool position and/or tool length offset)
I	I5.3 (I4.4)	Arc center modifier for X axis Shift amount in fixed cycles (X) Corner vector selection for axis (old type controls)
J	J5.3 (J4.4)	Arc center modifier for Y axis Shift amount in fixed cycles (Y) Corner vector selection for Y axis (old type controls)
K	K5.3 (K4.4)	Arc center modifier for Z axis
K (L)	K4 (L4)	Fixed cycle repetition count Subprogram repetition count
M	M2	Miscellaneous function (M3 for extended M-function set)
N	N5	Block number or sequence number (N4 for older controls)
O	O4 (O5)	Program number (EIA) or (:4 or :5 for ISO - not common)
P	P4	Subprogram number call Custom macro number call
	P3	Work offset number - used with G10
	P53	Dwell time in milliseconds
	P5	Block number in main program when used with M99

Milling System Format

Address descriptions vary for many addresses, depending on the input units. The table below lists metric format descriptions (imperial format is in parenthesis, if applicable). Listed are format notations for milling units.

Some addresses have additional meaning when used in special applications

The first column is the address, second column is the format notation, third column is the description:

Turning System Format

Similar chart as for milling, this one is for lathe systems. A number of definitions are the same and are included only for convenience. Notation is in the metric format, imperial notation is in parenthesis, if applicable to the address.

Some addresses have additional meaning when used in multiple repetitive cycles of the G70 series

or special applications

Address	Notation	Description
A	A3	Angle of thread for G76 Angle for direct drawing input
C	C 5.3 (C 4.4)	Chamfer for direct drawing input
	C 5.3	Absolute axis rotation in degrees
D	D4	Number of divisions in G73
	D53 (D44)	Depth of cut in G71 and G72, Relief amount in G74 and G75 Depth of first thread in G76
E	E3.5 (E2.6)	Precision feed rate for threading
F	F3.3 (F2.4)	Feed rate function - may vary
G	G2	Preparatory commands (G3 for extended G-code set)
H	H 5.3	Incremental axis rotation in degrees
I	I 5.3 (I 4.4)	Arc center modifier for X axis Taper height in X for cycles X axis relief in G73 Direction of chamfering Motion amount in X in G74
K	K 5.3 (K 4.4)	Arc center modifier for Z axis Taper height in Z for cycles Z axis relief in G73 Direction of chamfering Motion amount in Z in G75 Thread depth in G76
K (L)	K4 (L4)	Subprogram repetition count
M	M2	Miscellaneous function (M3 for extended M-function set)
N	N5	Block number or sequence number (N4 for older controls)
O	O4 (O5)	Program number (EIA) or (:4 or :5 for ISO - not common)
P	P4 (P5)	Subprogram number call Custom macro number call Offset number with G10 Return to block number with M99 Start block number in G71 and G72
	P53	Dwell time in milliseconds
Q	Q4 (Q5)	End block number in G71 and G72

R	R 5.3 (R 4.4)	Arc radius designation Arc radius for corners
S	S5	Spindle speed in r/min or ft/min
T	T4	Tool function
U	U 5.3 (U 4.4)	Incremental value in X axis Stock allowance in X axis
	U5.3	Dwell function with G04
W	W 5.3 (W 4.4)	Incremental value in Z axis Stock allowance in Z axis
X	X 5.3 (X 4.4)	Absolute value in X axis
	X5.3	Dwell function with G04 only
Z	Z 5.3 (Z 4.4)	Absolute value in Z axis

Multiple Word Addresses

One feature that is noticeable in both tables is the abundance of different meanings for some addresses. This is a necessary feature of the word address format. There are only twenty-six letters in the English alphabet but more than that the number of commands and functions. As new control features are added due to technological advancements, even more variations may be necessary in the future. Several addresses have such an established meaning (for example, X, Y and Z are coordinate words), that giving them any additional meaning would be confusing. Many letters, on the other hand, are not used very often and their multiple meaning is quite acceptable (typical addresses are I, J, K, P, Q, for example). In addition, the actual meaning of addresses varies between milling and turning systems and even between manufacturers.

The control system has to have some means of accepting a particular word with a precisely defined meaning in the program. In most cases, the preparatory command G will define its meaning, at other times it will be an M function or a setting of system parameters.

E. SYMBOLS IN CNC PROGRAMMING

In addition to the basic symbols, Fanuc can accept other symbols for different applications. The following table describes all symbols available on Fanuc and compatible control systems:

Symbol	Description	Comment
.	Decimal point	Fractional part of a number
+	Plus, sign	Positive value or <i>addition</i> sign in Fanuc macros
-	Minus sign	Negative value or <i>subtraction</i> sign in Fanuc macros
*	Multiplication sign	<i>Multiplication</i> sign in Fanuc macros
/	Slash (Front slash)	Block skip function symbol or <i>division</i> sign in Fanuc macros
()	Parenthesis	Program comments & messages
%	Percent sign	Stop code (end of program file)
:	Colon	Program number designation (rare)

,	Comma	Used only within comments
[]	Brackets	Calculations in Fanuc macros
;	Semicolon	Nonprogrammable End-Of-Block symbol (screen display only)
#	Sharp sign	Variable definition or call in Fanuc macros
=	Equal sign	Equality symbol in Fanuc macros

The table above lists typical standard and special symbols available on most control systems.

Special symbols are used only with optional features, such as *custom macro-option*. These symbols cannot be used in standard programming, as they would cause an error. Typical standard symbols are found on the computer keyboard. *Ctrl*, *Shift* and *Alt* character combinations are not allowed in CNC programming.

Plus, and Minus Sign

One of the most common symbols in mathematics is an algebraic sign - *plus* or *minus*. In CNC, any data in a motion command can be either positive or negative. Virtually all control systems allow for the omission of a plus sign for all positive values. This feature is sometimes called *positive bias* of the control system. This is a mathematical term indicating assumed positive value of a number, if no sign is programmed. In CNC, the implied positive sign is always *after* the address (letter) in a word:

X+125.0 *is the same as X125.0*

For negative numbers, the minus sign symbol must always be programmed. If minus sign is missing, the number automatically becomes positive, with an incorrect result value, in this case, a possible tool position:

X-125.0 *Negative value (-sign is necessary)*

X125.0 *Positive value (no sign means positive value)* **X+125.0** *Positive value (+sign is ignored)*

Symbols supplement letters and digits and are important part of CNC program structure. Not all symbols are used in every program - in fact, only very few are normally used.

F. TYPICAL PROGRAM STRUCTURE

(BLANK LINE MAY BE NECESSARY - OR % SYMBOL)

O0701 (ID MAX 15 CHARS) (PROGRAM NUMBER AND ID)

(SAMPLE PROGRAM STRUCTURE FOR FIXED CYCLES) (BRIEF PROGRAM DESCRIPTION)

(PETER SMID - 07-DEC-2008)

(PROGRAMMER AND DATE OF LAST REVISION)

(BLANK LINE)

N1 G21

(UNITS SETTING IN A SEPARATE BLOCK)

N2 G17 G40 G80 G49

(INITIAL SETTINGS AND CANCELLATIONS)

N3 T01

(TOOL T01 INTO WAITING POSITION)

N4 M06

(T01 INTO SPINDLE)

N5 G90 G54 G00 X.. Y.. S.. M03

(T01 RESTART BLOCK - T02 INTO WAITING

T02

POSITION)

N6 G43 Z25.0 H01 M08

(TOOL LG OFFSET - CLEAR ABOVE WORK -
COOLANT ON)

N7 G99 G82 X.. Y.. R.. Z.. P.. F.. (----	(FIXED CYCLE - G82 USED AS AN EXAMPLE)
CUTTING MOTIONS WITH TOOL	
T01 ----) ...	
N33 G80 Z25.0 M09	(CYCLE CANCEL - CLEAR ABOVE PART -
	COOLANT OFF)
N34 G28 Z25.0 M05	(HOME IN Z ONLY-SPINDLE OFF)
N35 M01	(OPTIONAL STOP)
	(--- BLANK LINE ---)
N36 T02	(TOOL T02 INTO WAITING POSITION - CHECK
	ONLY)
N37 M06	(T02 INTO SPINDLE)
N38 G90 G54 G00 X.. Y.. S.. M03	(T02 RESTART BLOCK - T03 INTO WAITING
T03	POSITION)
N39 G43 Z25.0 H02 M08	(TOOL LG OFFSET - CLEAR ABOVE WORK -
	COOLANT ON)
N40 G99 G81 X.. Y.. R.. Z.. F..	(FIXED CYCLE - G81 USED AS AN EXAMPLE)
(---- CUTTING MOTIONS WITH	
TOOL T02 ----) ...	
N62 G80 Z25.0 M09	(CYCLE CANCEL - CLEAR ABOVE PART -
	COOLANT OFF)
N63 G28 Z25.0 M05	(HOME IN Z ONLY - SPINDLE OFF)
N64 M01	(OPTIONAL STOP)
	(--- BLANK LINE ---)
N65 T03	(TOOL T03 INTO WAITING POSITION - CHECK
	ONLY)
N66 M06	(T03 INTO SPINDLE)
N67 G90 G54 G00 X.. Y.. S.. M03	(T03 RESTART BLOCK - T01 INTO WAITING
T01	POSITION)
N68 G43 Z25.0 H03 M08	(TOOL LG OFFSET - CLEAR ABOVE WORK -
	COOLANT ON)
N69 G99 G84 X.. Y.. R.. Z.. F..	(FIXED CYCLE - G84 USED AS AN EXAMPLE)
(---- CUTTING MOTIONS WITH	
TOOL T03 ----) ...	
N86 G80 Z25.0 M09	(CYCLE CANCEL - CLEAR ABOVE PART -
	COOLANT OFF)
N87 G28 Z25.0 M05	(HOME IN Z ONLY - SPINDLE OFF)
N88 G28 X.. Y..	(HOME IN XY ONLY - MAY BE OMITTED)
N89 M30	(END OF PROGRAM)
%	(STOP CODE - END OF FILE TRANSFER)

Although it may be a bit early to show a complete program, it will do no harm to look at a typical program structure. In the next two examples, a typical mill-type structure is shown - one for fixed cycles, one for other machining:

The XY value in the block N88 should be the *current position* of the X and Y axes. If the absolute position is unknown, change the block to the incremental version:

N88 G91 G28 X0 Y0

If a tool has to be repeated, make sure not to include tool change block for the current tool. Many CNC systems will generate an alarm if the tool change command cannot find a particular tool in the magazine. In both preceding examples, tool repeat blocks will be N5, N38 and N67.

The program structure examples are for a vertical CNC machining center with random tool

selection mode and typical Fanuc control system - some minor changes are to be expected. Study the *program flow* rather than its exact content. Note the repetitiveness of blocks for each tool and also note a blank line (empty block) between individual tools for easier orientation.

	(BLANK LINE MAY BE NECESSARY - OR % SYMBOL)
O0702 (ID MAX 15 CHARS)	(PROGRAM NUMBER AND ID)
(SAMPLE PROGRAM STRUCTURE FOR MILLING)	(BRIEF PROGRAM DESCRIPTION)
(PETER SMID - 07-DEC-2008)	(PROGRAMMER AND DATE OF LAST REVISION)
	(BLANK LINE)
N1 G21	(UNITS SETTING IN A SEPARATE BLOCK)
N2 G17 G40 G80 G49	(INITIAL SETTINGS AND CANCELLATIONS)
N3 T01	(TOOL T01 INTO WAITING POSITION)
N4 M06	(T01 INTO SPINDLE)
N5 G90 G54 G00 X.. Y.. S.. M03 T02	(T01 RESTART BLOCK - T02 INTO WAITING POSITION)
N6 G43 Z25.0 H01 M08	(TOOL LG OFFSET - CLEAR ABOVE WORK - COOLANT ON)
N7 G01 Z.. F..	(FEED TO Z CLEARANCE OR DEPTH)
(---- CUTTING MOTIONS WITH TOOL T01 ----) ...	
N33 G00 Z25.0 M09	(CLEAR ABOVE PART - COOLANT OFF)
N34 G28 Z25.0 M05	(HOME IN Z ONLY-SPINDLE OFF)
N35 M01	(OPTIONAL STOP)
	(--- BLANK LINE ---)
N36 T02	(TOOL T02 INTO WAITING POSITION - CHECK ONLY)
N37 M06	(T02 INTO SPINDLE)
N38 G90 G54 G00 X.. Y.. S.. M03 T03	(T02 RESTART BLOCK - T03 INTO WAITING POSITION)
N39 G43 Z25.0 H02 M08	(TOOL LG OFFSET - CLEAR ABOVE WORK - COOLANT ON)
N40 G01 Z.. F..	(FEED TO Z CLEARANCE OR DEPTH)
(---- CUTTING MOTIONS WITH TOOL T02 ----) ...	
N62 G00 Z25.0 M09	(CLEAR ABOVE PART - COOLANT OFF)
N63 G28 Z25.0 M05	(HOME IN Z ONLY - SPINDLE OFF)
N64 M01	(OPTIONAL STOP)
	(--- BLANK LINE ---)
N65 T03	(TOOL T03 INTO WAITING POSITION - CHECK ONLY)
N66 M06	(T03 INTO SPINDLE)
N67 G90 G54 G00 X.. Y.. S.. M03 T01	(T03 RESTART BLOCK - T01 INTO WAITING POSITION)
N68 G43 Z25.0 H03 M08	(TOOL LG OFFSET - CLEAR ABOVE WORK - COOLANT ON)
N69 G01 Z.. F..	(FEED TO Z CLEARANCE OR DEPTH)
(---- CUTTING MOTIONS WITH TOOL T03 ----) ...	

N86 G00 Z25.0 M09	(CLEAR ABOVE PART - COOLANT OFF)
N87 G28 Z25.0 M05	(HOME IN Z ONLY - SPINDLE OFF)
N88 G28 X.. Y..	(HOME IN XY ONLY - MAY BE OMITTED)
N89 M30	(END OF PROGRAM)
%	(STOP CODE - END OF FILE TRANSFER)

Each program block is identified with a comment as brief explanation. In practice, the two versions shown are combined in a single program. Actual meaning of the content will be explained in subsequent chapters in this digital book.

Program Structure Benefits

Developing a solid program structure is absolutely essential for one simple reason - *it is going to be used all the time*. Practical benefits are numerous - they include ease of reading the program (on screen or on paper), starting with any tool is easier, important default settings are defined *after* tool change (for example, G90, G54, G00). All these (and other) benefits result in fewer errors and increased productivity at the CNC machine.

Well-designed program structure is worth the time spent developing it

G. PROGRAM HEADER

Examples above showed a typical program structure for milling machines and machining centers. At the beginning of a typical program, there is usually a block or two of comments that provide general information. Various comments and/or messages may be placed in the program, providing they are enclosed in parentheses. This kind of internal documentation is useful to both the programmer and operator.

A special series of comments at the program *top* is defined as the *program header*, where various program features are identified. The next example is a rather much exaggerated sample of items that *may* be used in a program header:

```
(FILE NAME ..... O1234.NC)
(LAST VERSION DATE ..... 07-DEC-2008)
(LAST VERSION TIME ..... 19:43)
(PROGRAMMER ..... PETER SMID)
(MACHINE ..... OKK - VMC)
(CONTROL ..... FANUC 30i)
(UNITS ..... METRIC)
(JOB NUMBER ..... 4321) (OPERATION ..... DRILL-BORE-TAP)
(STOCK MATERIAL ..... H.R.S. PLATE)
(MATERIAL SIZE ..... 200 X 150 X 50)
(PROGRAM ZERO ..... X0 - LEFT EDGE)
(      Y0 - BOTT EDGE)
(      Z0 - TOP FACE)
(STATUS ..... NOT VERIFIED)
(-----)
```

Tool Comments

Some CNC programmers like to list all tools at the program beginning, also as comments. For example:

(=== T01 - 100 MM DIA FACE MILL =====)

(=== T02 - 25 MM DIA END MILL - 4 FLUTES ===)

(=== T08 - 10 MM SPOT DRILL =====)

(=== T15 - 7 MM DRILL - LONG SERIES =====)

Within the program, each tool may be identified as well, using the same comment at the beginning of each tool:

(=== T08 - 10 MM SPOT DRILL =====)

N15 T08

N16 M06

N17 G90 G54 G00 X.. Y.. S.. M03 T15 N18 ...

Other comments and messages to the operator can be added to the program as required, such as operation description or special action required.

7. PREPARATORY COMMANDS

The program address G identifies a *preparatory command*, often called a *G-code*. This address has one and only objective - that is *to preset* or *to prepare* the control system to a certain desired *condition*, or to a certain *mode* or a *state* of operation. For example, address G00 presets a rapid motion mode for the machine tool but does not move any axis, address G81 presets the drilling cycle but does not drill any holes, etc. The common term *preparatory command* indicates its meaning - a G-code will *prepare* control unit to accept the programming instructions *following* the G-code, in a specific way.

A. DESCRIPTION AND PURPOSE

One block example will illustrate the *need for* preparatory commands in the following program entry:

N7 X13.0 Y10.0

Even a casual look at this block shows that the coordinates X13.0Y10.0 relate to the cutting tool *end position*, when block N7 is executed (*i.e.*, processed by the control). Block N7 does not indicate whether the coordinates are in absolute or incremental mode. It does not indicate whether X13.0Y10.0 are in metric or imperial units. Neither it indicates whether the actual motion to this specified target position is a rapid motion or a linear motion. If evaluation of such a block cannot establish the meaning of its contents, control system does not have enough information. The supplied information in such a block is *incomplete*, therefore unusable by itself. Some additional instructions for the coordinates are required that define their full purpose.

For example, in order to make the block N7 a tool motion in *rapid* mode (G00) using *absolute dimensions* (G90), *all* these instructions - or commands - must be specified *before* the block or *within* the block:

Example A:

N7 G90 G00 X13.0 Y10.0 Example B:

N3 G90

N4 ...

N5 ...

N6 ...

N7 G00 X13.0 Y10.0

Example C:

N3 G90 G00

N4 ...

N5 ...

N6 ...

N7 X13.0 Y10.0

Example D:

N2 G90

N3 G00

N4 ...

N5 ...

N6 ...

N7 X13.0 Y10.0

All four examples have the same machining result, providing that there is no change of any G-code mode between blocks N4 and N6 in the examples B, C and D. G-codes that can be written once and stay in effect until canceled or changed are *modal* G-codes, divided into logical groups.

One G-code in a modal group replaces another G-code from the same group

Modal and non-modal G-codes will be described shortly. Each control system has its own list of available G-codes. Many G-codes are very common and can be found on virtually all controls, others are unique to the particular control system, even the machine tool. Because of the nature of machining applications, the list of typical G-codes will be different for milling systems and turning systems. The same applies for other types of machines. Each group of G-codes must be kept separate.

Check machine documentation for available G codes!

B. APPLICATIONS FOR TURNING

Fanuc lathe controls use three G-code group types A, B and C. *Type A* is the most common; in this digital book, all examples and explanations are *Type A* group, including the table below. Only one type can be set at a time. Types A and B can be set by control system parameter, but type C is optional. Generally, most G-codes are identical, only a few are different in the A and B type groups.

Most of the preparatory commands are discussed under their individual applications, for example, G01 will be explained under *Linear Interpolation*, G02 and G03 under *Circular Interpolation*, etc. In this section, G-codes are described in general, regardless of the type of machine or control unit.

C. G-CODES IN A PROGRAM BLOCK

Unlike miscellaneous functions, known as the *M-functions* and described in the next chapter, several preparatory commands may be used in a single block, providing they are not in a logical conflict with each other:

N25 G90 G00 G54 X6.75 Y10.5

This method of program writing is several blocks shorter than its single block alternative:

N25 G90

N26 G00

N27 G54

N28 X6.75 Y10.5

Both methods will be identical during continuing processing. However, the second example, when executed in a *single block* mode, each block will require pressing the *Cycle Start* key to activate each block. The shorter method is more practical, not only for its length, but for the logical connection between individual addresses within the block.

Some rules of application and general considerations apply to G-codes when used with *other* data in a block. The most important of them is the subject of *modality*.

Modality of G-codes

Earlier, the following example C was used to demonstrate the general placement of several G-codes into a program block:

Example C - original:

N3 G90 G00

N4 ...

N5 ...

N6 ...

N7 X13.0 Y10.0

If the structure is changed slightly and filled with realistic data, these five blocks may be the result:

Example C - modified (as programmed):

N3 G90 G00 X5.0 Y3.0

N4 X0

N5 Y20.0

N6 X15.0 Y22.0

N7 X13.0 Y10.0

Note the rapid motion command G00 - how many times does it appear in the program? Just *once* - in block N30. In fact, so does the command for absolute mode, G90 - only once. The reason neither G00 nor G90 has been repeated in every block is because both commands remain active from the moment of their first appearance in the program. The term *modal* is used to describe this characteristic.

For a command to be modal, it has to remain in a certain mode until canceled by an equivalent mode

As most (not all) G-codes are modal, there is no need to repeat a modal command in every block. Using the earlier example C once more, control system will make the following interpretation during program execution:

Example C - modified (as processed):

N3 G90 G00 X50.0 Y30.0

N4 G90 G00 X0

N5 G90 G00 Y200.0

N6 G90 G00 X150.0 Y220.0

N7 G90 G00 X130.0 Y100.0

This program example does not represent any practical application by moving from one location to another at a rapid rate, but it demonstrates the modality of preparatory commands. The main purpose of modal values is to avoid unnecessary duplication of programming modes. G-codes are used so often, that writing them in the program can be tedious. Fortunately, the majority of G-codes can be applied only once, *providing they are modal*. In the control system specifications, preparatory commands are identified as modal and unmodal.

Conflicting Commands in a Block

Preparatory commands are used in the program, to select from two or more possible modes of operation. If the rapid motion command G00 is selected, it is a specific command relating to a tool motion. As it is impossible to have a rapid motion and a cutting motion active *at the same time*, it is impossible to have G00 and G01 active at the same time. Such a combination creates a *conflict* in a block. If conflicting G-codes are used in the same block, the *latter* G-code will be used by the control (system error is also possible).

N74 G01 G00 X3.5 Y6.125 F20.0

In the example, the two commands G01 and G00 are in conflict. As G00 is the *latter* one in the block, it will become effective. The feed rate is ignored in this case.

N74 G00 G01 X3.5 Y6.125 F20.0

This is the exact opposite of the previous example. Here, the G00 is first, therefore G01 command will take precedence and the motion will take place as a cutting motion at the specified feed rate of 20.0 in/min.

Word Order in a Block

G-codes are normally programmed *at the beginning of a block*, after the block number, before other significant data:

N40 G91 G01 Z-0.625 F8.5

This is a traditional order, based on the idea that if the purpose of G-codes is to *prepare* or *preset* control system to a certain condition, the preparatory commands should always be placed first. Supporting this argument is the fact that only non-conflicting codes are allowed in a single block. Strictly speaking, there is nothing wrong with rearranging the order to:

N40 G91 Z-0.625 F8.5 G01

Perhaps unusual, but quite correct. That is not the case with the next method of positioning a G-code in a block:

N40 Z-0.625 F8.5 G01 G91

Watch for situations like this! What happens in this case is that the cutting motion G01, feed rate F and depth Z will be combined and executed *using the current dimensional mode*. If the current mode is absolute, Z-axis motion will be executed as an absolute value, *not* an incremental value. The reason for this exception is that Fanuc allows to mix dimensional values in the *same* block. That can be a very useful feature, if used carefully. A typical *correct* application of this feature can be illustrated in this example:

(G20)

N45 G90 G00 G54 X1.0 Y1.0 S1500 M03 (G90)

N46 G43 Z0.1 H02

N47 G01 Z-0.25 F5.0

N48 X2.5 G91 Y1.5 (G90 MIXED WITH G91)

N49

Blocks N45 through N47 are all in absolute mode. Before the block N48 is executed, absolute position of the axes X and Y is 1.0,1.0. From this starting position, the target location is the absolute position of X2.5 combined with the incremental motion of 1.5 inches along the Y-axis. The resulting absolute position will be X2.5Y2.5, making a 45° motion. In this case, G91 will remain in effect for *all* subsequent blocks, until G90 is programmed. Most likely, the block N48 will be written in absolute mode (still active):

...

N48 X2.5 Y2.5

...

Normally, there is no reason to switch between the two modes in the same block. It can result in some very unpleasant surprises, and some controls may not support this method. Yet, there are some occasions when this special technique brings benefits, for example, in subprograms.

D. GROUPING OF COMMANDS

Earlier example of conflicting G-codes in a single block brings one issue to the forefront. It makes sense, for example, that motion commands such as G00, G01, G02 and G03 cannot co-exist in the same block. Such distinction is not so clear for other preparatory commands. For example, can the tool length offset command G43 be programmed in the same block as the cutter radius offset command G41 or G42 - even in theory only? The answer is yes, but let's look at the reason why.

Fanuc control system recognizes preparatory commands by separating them into arbitrary *groups*. Each group, called the *G-code group*, has a Fanuc assigned arbitrary two-digit number. The main rule governing the co-existence of G-codes in a single block is very simple. If two or more G-codes from *the same group* are programmed in the same block, they are in conflict with each other.

Group Numbers

G-code groups are typically numbered from 00 to 25. This range varies between different control models, depending on the features. It can even be higher for the newest controls or where more G-codes are offered by the machine control. One of these groups - the most unique one and perhaps the most important as well - is the *Group 00*.

All preparatory commands in Group 00 are *not* modal, sometimes using the descriptions *unmodal* or *non-modal*. They are only active in the block in which they were programmed. If unmodal G-codes are to be effective in several consecutive blocks, they *must* be programmed in *each* of those blocks. For majority of unmodal commands, this repetition will not be used very often.

For example, a dwell is a programmed pause measured in milliseconds. It is needed only for the duration within specified time, no longer. There is no logical need to program dwell in two or more consecutive blocks. After all, what is the benefit of the next three blocks?

N56 G04 P2000

N57 G04 P3000

N58 G04 P1000

All three blocks contain the same function - a dwell - one after another. The part program can be made much more efficient by simply entering the total dwell time into a single block with identical results:

N56 G04 P6000

The following groups are typical for Fanuc control systems. Applications for milling and turning controls are specially distinguished by the *M* and *T* letters respectively, in the *Type* column of the table:

Group	Description	G codes	Type
00	Unmodal G codes	G04	M/T
		G09 G10 G11	M/T
		G27 G28 G29 G30 G31 G37	M/T
		G45 G46 G47 G48	M/T
		G52 G53 G65	M/T
		G51 G60 G92	M
		G50	T
		G70 G71 G72 G73	T
		G74 G75 G76	T

01	Motion Commands, Cutting Cycles	G00 G01 G02 G03 G32 G35 G36 G90 G92 G94	M/T T T
02	Plane Selection	G17 G18 G19	M
03	Dimensioning Mode	G90 G91 (U and W for lathes)	M T
04	Stored Strokes	G22 G23	M/T
05	Feed rate	G93 G94 G95	T
06	Units Input	G20 G21	M/T
07	Cutter Radius Offset	G40 G41 G42	M/T
08	Tool Length Offset	G43 G44 G49	M
09	Cycles	G73 G74 G76 G80 G81 G82 G83 G84 G85 G86 G87 G88 G89	M M M M
10	Return Mode	G98 G99	M
11	Scaling Cancel, Mirror Image	G50 G68 G69	M T
12	Coordinate System	G54 G55 G56 G57 G58 G59	M/T M/T
13	Cutting Modes	G61 G62 G64 G63	M/T M
14	Macro Mode	G66 G67	M/T
16	Coordinate Rotation	G68 G69	M
17	CSS	G96 G97	T
18	Polar Input	G15 G16	M
24	Spindle Speed Fluctuation	G25 G26	M/T

The group relationship makes a perfect sense in all cases. One possible exception is Group 01 for *Motion Commands* and Group 09 for *Cycles*. The relationship between these two groups is this - if a G-code from Group 01 is specified in any of the fixed cycle Group 09, the cycle is immediately canceled, *but the opposite is not true*. In other words, an active motion command is *not* canceled by a fixed cycle.

Group 01 is not affected by G-codes from Group 09. In a summary ...

Any G-code from a given group automatically replaces another G-code from the same group

E. G-CODE TYPES

Fanuc control system offers a flexible selection of preparatory commands. This fact distinguishes Fanuc from many other controls. Considering the fact that Fanuc controls are used

worldwide, it only makes sense to allow the standard control configuration to follow established style of each country. A typical example is the selection of dimensional units. In Europe, Japan and many other countries, metric system is the standard. In North America, the common system of dimensioning still uses imperial units. As both markets are substantial in the world trade, a clever control manufacturer tries to reach them both. Almost all control manufacturers offer a selection of the dimensional system. But Fanuc and similar controls also offer selection of programming codes that were in effect *before* Fanuc reached the worldwide market.

The method Fanuc controls use is a simple method of parameter setting. By selecting a specific system parameter, one of two or three G-code types can be selected, the one that is typical for a particular geographical user. Although the majority of G-codes are the same for every type, the most typical illustration are G-codes used for imperial and metric selection of units. Many earlier US controls used G70 for imperial units and G71 for metric units. Fanuc system has traditionally used G20 and G21 codes for imperial and metric input respectively.

Setting up a parameter, the G-code type that is the most practical can be selected. Such a practice, if done at all, should be done only once and only when the control is installed, before any part programs have been written for it. Change of the G-code type at random is a guaranteed way to create organizational nightmare. Keep in mind that a change of one code meaning will affect the meaning of another code. Using the units example for a lathe, if G70 means an imperial input of dimensions, you cannot use it to program a roughing cycle. Fanuc provides a different code in that group. Always stay with the standard G-code type. All G-codes in this digital book use the default group of the *Type A*, and also the most common group.

G-Codes and Decimal Point

Many latest Fanuc controls include a G code with a decimal point, for example, G72.1 (Rotation copy) or G72.2 (Parallel copy). Several preparatory commands in this group are related to a particular machine tool or are not typical enough to be described in this digital book.

8. MISCELLANEOUS FUNCTIONS

The address *M* in a CNC program identifies a *miscellaneous function*, sometimes called a *machine function*. Not all miscellaneous functions are related to the operation of a CNC machine - quite a few are related to the processing of part program. The more suitable term *miscellaneous functions* are used throughout this digital book.

A. DESCRIPTION AND PURPOSE

Within the structure of a CNC program, programmers often need some means of activating certain aspects of machine operation or controlling the program flow. Without availability of such means, any part program would be incomplete and impossible to run. First, let's look at those miscellaneous functions that relate to the operation of a CNC machine - the true *machine functions*.

Machine Related Functions

Various physical operations of a CNC machine must be controlled by the program, to ensure fully automated machining. These functions generally use the M-address and include the following operations:

Spindle rotation	CW or CCW
Gear range change	Low / Medium / High
Automatic tool change	ATC
Automatic pallet change	APC
Coolant operation	ON or OFF
Tailstock or quill motion	IN or OUT

These operations vary between machines, due to the different designs by various machine manufacturers. A machine design, from the engineering point of view, is based on a certain primary machining application. A CNC milling machine will require different functions related to this type of machine than a CNC machining center or a CNC lathe. A numerically controlled EDM wire cutting machine will have many special functions, typical to that kind of machining, including those not found on other machines.

Even two machines designed for the *same type* of work, for example, two kinds of CNC vertical machining centers, will have functions different from each other, if they have a different control system or significantly different options. Various machine models from the same manufacturer will also have certain unique functions, even if the CNC systems are identical.

All machine tools designed for metal removal by cutting have certain common features and capabilities. For example, spindle rotation can have three - *and only three* - possible selections in a program:

Spindle normal rotation

Spindle reverse rotation, Spindle stop

In addition to these three possibilities, there is a function called the spindle orientation, also a machine related function. Another example is coolant. Coolant can only be controlled in a program as being either ON or OFF.

These operations are typical to most CNC machines. All are programmed with an M-function, typically followed by two digits, although some control models allow the use of a three-digit M-function, Fanuc 16/18, for example.

Fanuc also uses three-digit M-functions in several special applications, for example, for synchronization of two independent turrets on a multi-axis lathe. All these and other functions are related to the operation of CNC machines and belong to the group collectively known as *miscellaneous functions* or simply as the *M-functions* or *M-codes*.

Program Related Functions

In addition to the machine functions, several M-functions are used to control the execution of a CNC program. An interruption of a program execution requires an M-function, for example, during the change of a job setup, such as part reversal. Another example is a situation where one program calls one or more subprograms. In such a case, each program has to have a program call function, the number of repetitions, etc. M-functions handle these requirements.

Based on previous examples, using miscellaneous functions falls into two main groups, based on a particular application:

Control of machine functions

Control of program execution

This digital book covers only the most common miscellaneous functions, used by the majority of controls. Unfortunately, there are many functions that vary between machines and control systems. These functions are called *machine specific functions*. For this reason, always consult documentation for the particular machine model and its control system.

B. TYPICAL APPLICATIONS

Before learning any M-functions, note the type of activity these functions do, regardless of whether such activity relates to the machine or the program. Also note the abundance of two-way toggle modes, such as *ON* and *OFF*, *IN* and *OUT*, *Forward* and *Backward*, etc. Always check your manual first - for reasons of consistency, all M-functions in this digital book are based on the following table:

Applications for Turning

Function	Description
M00	Compulsory program stop
M01	Optional program stop
M02	End of program (usually with reset, no rewind)
M03	Spindle rotation normal (CW for R/H tools)
M04	Spindle rotation reverse (CCW for R/H tools)
M05	Spindle stop
M07	Coolant mist ON <i>(usually an option)</i>
M08	Coolant ON (coolant pump motor ON)
M09	Coolant OFF (coolant pump motor OFF)
M10	Chuck open
M11	Chuck close
M12	Tailstock quill IN
M13	Tailstock quill OUT
M17	Turret indexing forward
M18	Turret indexing reverse

M19	Spindle orientation (optional)
M21	Tailstock forward
M22	Tailstock backward
M23	Thread gradual pull-out ON
M24	Thread gradual pull-out OFF
M30	Program end (always with reset and rewind)
M41	Low gear selection
M42	Medium gear selection 1
M43	Medium gear selection 2
M44	High gear selection
M48	Feed rate override cancel OFF <i>(deactivated)</i>
M49	Feed rate override cancel ON <i>(activated)</i>
M98	Subprogram call
M99	Subprogram end

Special MDI Functions

Several M-functions cannot be used in a CNC program at all. This group is used in so called *Manual Data Input* mode exclusively (MDI). An example of such functions is a step-by-step tool change for machining centers, used for service purposes only, never in the program. These functions are outside the scope of this digital book.

Application Groups

The two major categories, described earlier, can further be divided into several groups, based on the specific application of the miscellaneous functions within each group. A typical distribution list is contained in the following table:

Group	Typical M-functions
Program	M00 M01 M02 M30
Spindle	M03 M04 M05 M19
Tool change	M06
Coolant	M07 M08 M09
Accessories	M10 M11 M12 M13 M17 M18 M21 M22 M78 M79
Threading	M23 M24

Gear ranges	M41 M42 M43 M44
Feed rate override	M48 M49
Subprograms	M98 M99
Pallets	M60

This table does not cover all M-functions or even all possible groups. Neither does it distinguish between machines. On the other hand, it does indicate the *types* of applications miscellaneous functions are used for in everyday CNC programming.

Miscellaneous functions listed in this chapter are used throughout the book. Some of them appear more often than others, reflecting their general use in programming. Functions that do not correspond to a particular machine control system are either not used or not needed. However, the concepts for their applications are always similar for most control systems and CNC machines.

In this chapter, only the more general functions are covered in significant detail. Remaining miscellaneous functions are described in the sections covering individual specific applications. At this stage, the stress is on usage and behavior of the most common miscellaneous functions.

C. M-FUNCTIONS IN A BLOCK

If a miscellaneous function is programmed in a block by itself, with no other data supplementing it, only the function itself will be executed. For example,

N45 M01

is an *optional stop*. This block is correct - an M-function *can* be the only block entry. Unlike the preparatory commands (G codes), only *one* M-function is allowed in a block - unless the control allows multiple M-functions in the same block (some latest controls only), a program error will occur.

More practical method of programming certain miscellaneous functions is in a block that contains a *tool motion*. For example, turning the coolant on and - *at the same time* moving the cutting tool to a certain part location may be required. As there is no conflict between these instructions, the actual program block may look something like this:

N56 G00 X252.95 Y116.47 M08

In this example of block N56, the precise *time* the M08 function will be activated is not very important. In other cases, the timing may be very important. Some M-functions *must* be in effect before or after certain action takes place. For example, look at this combination - a Z-axis motion is applied together with *program stop* function M00 in the same block:

N319 G01 Z-62.5 F200.0 M00

This is a far more serious situation, and two answers are needed. One is *what exactly* will happen, the other is *when exactly* it will happen, when the M00 function is activated. There are three possibilities and three questions to ask:

Will the program stop take place immediately, when the motion is activated - at the start of block?

Will the program stop take place while the tool is on the way - during a motion?

Will the program stop take place when the motion command is completed - at the end of block?

One of the three options *will* happen - but which one? Even if a practical purpose of these examples may not be apparent at this stage, it is useful to know how the CNC system interprets blocks containing tool motion combined with a miscellaneous function.

Each M-function is logically designed - it also shows a great deal of *common sense*.

The actual startup of any M-function should be divided into *two* groups - not three, as the questions might suggest:

M-function activates at the start of block (simultaneously with the tool motion)

M-function activates at the end of block

(When the tool motion has been completed)

No M-function will be activated *during* block execution, there is no logic to it. What is the logical startup of coolant ON function M08 in block N56 above? The correct answer is that the coolant will be activated at the *same time* as the tool motion *begins*. The correct answer for example block N319 is that the M00 program stop function will be activated *after* the tool motion has been completed. Makes sense? Yes, but what about the other functions, how do they behave in a block with motion?

Let's look at them next.

Startup of M-Functions

Take a look at the list of typical M-functions. Add a tool motion to each and try to determine how the function is going to behave, based on the previous notes. A bit of logical thinking provides a good chance to arrive at the right conclusion. Compare the two following groups to confirm:

M-functions activated at the START OF A BLOCK	
M03	Spindle rotation normal (CW for R/H tools)
M04	Spindle rotation reverse (CCW for R/H tools)
M06	Automatic tool change (ATC)
M07	Coolant mist ON <i>(usually an option)</i>
M08	Coolant ON (coolant pump motor ON)
M-functions activated at the END OF A BLOCK	
M00	Compulsory program stop
M01	Optional program stop
M02	End of program (usually with reset, no rewind)
M05	Spindle stop
M09	Coolant OFF (coolant pump motor OFF)
M30	Program end (always with reset and rewind)
M60	Automatic pallet change (APC)

If there is any uncertainty about how an M-function will interact with tool motion, the safest choice is to program the M-function as a *separate block*. That way, it will always be processed *before* or *after* the relevant program block. For majority of applications this is a safe solution.

Duration of M-Functions

Knowledge of *when* an M-function takes effect is logically followed by another question - *how long* such a function will be active. Some miscellaneous functions are active only in the block they

appear. Others will continue to be in effect until canceled (replaced) by another miscellaneous function. This is similar to the modality of preparatory commands (G-codes); however, the word *modal* is not usually used with M-functions. As an example of a function duration, take miscellaneous functions M00 or M01. Either one will be active for *one* block only. The coolant ON function M08, will be active until a *canceled* or an *altering* function is programmed. Keep in mind that any one of the following functions will also cancel coolant ON mode - M00, M01, M02, M09 and M30. Compare these two tables:

M-functions completed in ONE BLOCK	
M00	Compulsory program stop
M01	Optional program stop
M02	End of program (usually with reset, no rewind)
M06	Automatic tool change (ATC)
M30	Program end (always with reset and rewind)
M60	Automatic pallet change (APC)
M-functions active UNTIL CANCELED or ALTERED	
M03	Spindle rotation normal (CW for R/H tools)
M04	Spindle rotation reverse (CCW for R/H tools)
M05	Spindle stop
M07	Coolant mist ON <i>(usually an option)</i>
M08	Coolant ON (coolant pump motor ON)
M09	Coolant OFF (coolant pump motor OFF)

Above classification is logical and shows good design and common sense. There is no need to remember each M-function and its exact activities. The best place to find out for certain, is to study manuals supplied with the CNC machine and watch the program run right on the machine.

D. PROGRAM FUNCTIONS

Miscellaneous functions that control program processing can be used either to interrupt the processing temporarily (in the middle of a program) or permanently (at the end of a program). Several functions are available for this purpose.

Program Stop

M00 function is defined as an *unconditional* or *compulsory* program stop. Any time CNC system encounters this function during program processing, all automatic operations of the machine tool will stop:

Motion of all axes

Rotation of the spindle

Coolant function

Further program execution

Control settings will *not* be reset when M00-function is processed. All significant program data currently active are still retained (feed rate, coordinate setting, spindle speed, etc.). Program processing can only be resumed by activating the *Cycle Start* key. M00 function cancels both spindle rotation and coolant function - either one or both have to be reprogrammed in subsequent blocks.

Miscellaneous function M00 can be programmed as an individual block or in a block containing other commands, usually axis motion. If M00 is programmed together with a motion command, the motion will be completed first, *then* the program stop will become effective:

M00 programmed *after a motion command*:

N38 G00 X189.5 N39 M00

M00 programmed *with a motion command*:

N39 G00 X189.5 M00

In both cases, any motion command will be completed first, *before* the program stop is executed. Actual difference between the two examples is apparent only in single block processing mode (for example, during a trial cut). There will be no practical difference in auto mode of processing (*Single Block* switch set to OFF position).

Practical Usage

Program stop function used in a program makes the CNC operator's job much easier. It is useful for many jobs. One common use is a part inspection at the machine, while the part is still mounted. In program stop mode, the part dimensions or tool condition can be checked. Chips accumulated in a bored or drilled hole can be removed, for example, before another machining operation can start. Blind hole tapping is a good example. Program stop function is also necessary to change the current setup before the program is completed, for example, to reverse a part. Any manual tool change also requires M00 function in the program.

Program stop function M00 is used only for a manual intervention during program processing

All control systems also offer an *optional* program stop M01, described next. The main rule of using M00 is the need of a *manual intervention* for *every* part machined. Manual tool change in a program qualifies for M00, because every part needs it. A dimensional check may not qualify, if is infrequent. In this case, M01 function will be a better choice. Although the difference between both functions is slight, the actual difference in cycle time can be significant for large number of parts.

When using M00 in a program, always inform the operator *why* the function has been used and what its purpose is. Make your intent known to avoid a confusion. This intent can be made available to the operator in two ways:

In setup sheet, refer to the block number that contains miscellaneous function M00 and describe any manual operation that has to be performed:

BLOCK N39 REMOVE CHIPS

In the program itself, issue a comment section with the necessary information. Comment section must be enclosed in parentheses (three versions shown):

N39 M00 (REMOVE CHIPS)

N39 X189.5 M00 (REMOVE CHIPS)

N39 X189.5 M00

(REMOVE CHIPS)

Any one of these methods will give the CNC operator all necessary information. From all options, the comment section [A] or [B] in a program is preferable. The built-in instructions can be read at the control panel display screen.

Optional Program Stop

Miscellaneous function M01 is an *optional, conditional* program stop. It is similar to M00 function, with one difference. Unlike M00 function, when M01 function is read by the control,

program processing will *not* stop without operator's interference. The *Optional Stop* toggle switch or a button key located on the operation panel can be set to either ON or OFF position. When M01 function in the program is processed, current switch setting will determine whether the program will temporarily stop, or processing continues without interruption:

Optional Stop switch setting	Result of M01
ON	Processing <i>will</i> stop
OFF	Processing <i>will not</i> stop

In case there is no M01 function programmed, the setting of the *Optional Stop* switch is not important. Normally, it should be set to OFF position for production work.

When active, M01 function behaves exactly as the M00 function. Motion of all axes, spindle rotation, coolant function and any further program execution will be temporarily interrupted. Feed rate, coordinate settings, spindle speed setting, etc., are retained. Further processing of the program can only be resumed by pressing the *Cycle Start* key. All programming rules for M00 function also apply to M01 function.

A good idea is to program M01 as the only entry in the last block of each tool, followed by a blank line with no data. If there is no need for program interruption, the *Optional Stop* switch will be set to OFF position and no production time is lost. If there is a need to stop processing at the end of a tool, the switch will be set to ON position and program processing stops when M01 is processed. Any time loss is usually justified under the circumstances, for example, to change a cutting insert or to inspect important dimensions or surface finish quality.

Program End

Every program must include a special function defining the *end* of active program. For this purpose, there are two M-functions available - the M02 and M30. Both are similar, but each has a distinct purpose. M02 function will terminate the program but *will cause no return* to the first block at the program top. Function M30 will terminate the program as well but it *will cause a return* to the program top. The word '*return*' is often replaced by the word '*rewind*'. It is a leftover from the times when a reel-to-reel tape reader was common on NC machines. The tape had to be *rewound* when the program has been completed for each part. M30 function provided this *rewind* capability.

When control system reads the program end function M02 or M30, it cancels all axis motions, spindle rotation, coolant function and usually resets the system to the default conditions. *On some controls the reset may not be automatic, and any programmer should be aware of it.*

If the program ends with M02 function (usually old programs only), the control remains at the program end, ready for the next *Cycle Start*. On modern CNC equipment there is no need for M02 at all, except for backward compatibility. This function was used in addition to M30 for those machines (mainly NC lathes) that had tape readers without reels, using a short loop tape. The tape trailer was spliced to the tape leader, creating a closed loop. When the program was finished, the start of tape was next to its end, so no rewind was necessary. Long tapes could not use loops and required reels and M30. So much for the history of M02 - just ignore its existence.

Is M02 the Same as M30?

On most modern controls, a system parameter can be set to make the M02 function with the same meaning as that of M30. This setting can give it the rewind capabilities, useful in situations where an old program can be used on a machine with a new control without changes.

In a summary, if the end of program is terminated by M30 function, the rewind *will* be performed; if the M02 function is used, the rewind *will not* be performed.

When writing a program, make sure the last block in the program contains nothing else but M30 as the preferred end (sequence block is allowed to start the block):

N65 . . .

N66 G91 G28 X0 Y0

N67 M30 (END OF PROGRAM)

%

On some controls, the M30 function can be used together with the axes motion - *definitely NOT recommended!*

N65 . . .

N66 G91 G28 X0 Y0 M30 (END OF PROGRAM)

%

Percent Sign

The percent sign (%) after M30 is a special *stop code*. This symbol terminates the loading of a program from an external device. It is also called the *end-of-file marker*.

Subprogram End

The last M-function for a program end is M99. Its primary usage is in subprograms. Typically, the M99 function will terminate a subprogram and return to the processing of the previous program. If M99 is used in a standard program, it creates a program with *no end* - such a situation is often called an *endless program loop*. M99 should be used only in subprograms, not in the standard programs.

E. MACHINE FUNCTIONS

Miscellaneous functions relating to the operation of machine tool are part of another group. This section describes the most important of them in detail.

Coolant Functions

Many metal removal operations require that the cutting tool is flooded with a suitable coolant. To control the flow of coolant in the program, there are three miscellaneous functions usually provided for this purpose:

M07	Mist ON
M08	Flood ON
M09	Mist or Flood OFF

Mist is the combination of a small amount of cutting oil mixed with compressed air. It depends on the machine tool manufacturer whether this function is standard for a particular CNC machine tool or not. Some manufacturers replace the mixture of oil and air with air only, or with oil shot only, etc. In these cases, it is typical that an additional equipment is built into the machine. If this option exists on the machine, the most common miscellaneous function to activate the oil mist or air is M07.

Function similar to M07 is M08, coolant *flood*. This is by far the most common method in CNC programming. It is standard for virtually all CNC machines. Coolant, usually a suitable mixture of soluble oil and water, is premixed and stored in the machine coolant tank. Some machines also need an M-function for coolant *through the spindle*. Flooding the tool cutting edge is important for three reasons:

Heat dissipation

Chip removal

Lubrication

Primary reason to use coolant flood aimed at the cutting edge is to dissipate the heat generated there during cutting. The secondary reason is to remove chips from the cutting area, using coolant pressure. Finally, the coolant also acts as a lubricant to ease the friction between cutting tool and material. Lubrication helps to extend tool life and improves surface finish.

During initial tool approach towards the part or during final return to the tool change position, coolant is normally not required. To turn off the coolant function, use the M09 function - *coolant off*. M09 will turn off supply from coolant tank and nothing else. In reality, the M09 function will shut off the coolant pump motor.

Each coolant related function may be programmed in separate blocks or together with an axis motion. There are subtle but important differences in the order and timing of program processing. Following examples explain the main differences:

Example A - oil mist is turned ON, if available:

N110 M07

Example B - coolant is turned ON:

N340 M08

Example C - coolant is turned OFF:

N500 M09

Example D - axis motion and coolant ON:

N230 G00 X11.5 Y10.0 M08

Example E - axis motion and coolant OFF:

N400 G00 Z1.0 M09

All examples show the differences in program processing. The general rules of coolant programming are:

Coolant ON or OFF in a *separate* block becomes active in the block in which it is programmed (Examples A, B and C)

Coolant ON, when programmed *with axes motion*, becomes active simultaneously with the axes motion

(Example D)

Coolant OFF, programmed *with axes motion*, becomes effective only upon completion of the axes motion (Example E)

The main purpose of M08 function is *to turn the coolant pump motor on*. It does not guarantee that the cutting edge receives any coolant immediately. On large machines with long coolant pipes, or machines with low coolant pump pressure, some delay is to be expected before coolant covers the distance between pump and cutting tool.

Coolant should always be programmed with two important considerations in mind:

There will be no coolant splashing outside of the work area (outside of the machine)

There will never be a situation when the coolant reaches a hot edge of the tool

The first consideration is relatively minor. If the coolant function is programmed in the 'wrong' place, the result may be just an inconvenience. Wet area around the machine may present unsafe working conditions and should be quickly corrected. Even more serious situation happens when

the coolant suddenly starts flooding a cutting tool that has already entered the material. The change in temperature at the cutting edge may cause the tool to break and damage the part. Carbide tools are far more easily affected by temperature changes than high speed steel tools. Such a possibility can be prevented during programming, by using the M08 function *a few blocks ahead* of the actual cutting block. Long pipes or insufficient coolant pressure on the machine may delay the start of the actual flooding.

Spindle Functions

Beginning on *page 81*, details of all aspects of controlling the machine spindle in a CNC program are covered. Miscellaneous functions that are available for spindle control are *spindle rotation* and *spindle orientation*.

Most machine spindles can rotate in both directions, *clockwise* (CW) and *counterclockwise* (CCW). The direction of rotation is always relative to a standard point of view. The viewpoint is established from the spindle side as the *direction along the spindle center line towards its face*. CW rotation in such a view is programmed as M03, CCW direction as M04, assuming the spindle can be rotated either way.

Both drilling and milling types of machines use this established convention quite commonly. The same convention is also applied to CNC lathes. On a CNC milling machine or a machining center, it is more practical to look towards the part from the spindle side rather than from the table side. On a lathe (slant bed horizontal type), the more practical view is *from the tailstock towards the spindle*, because that is the closest to how the CNC machine operator stands in front of the lathe. However, M03 and M04 spindle directions are established the same way as for machining centers. A further complication is the fact that left hand tools are used in lathe work more frequently than in the work for milling applications. Make an effort to study the instruction manual for a specific machine carefully - also see related details described on *page 82*.

Spindle function to program a spindle stop is M05. This function will stop the spindle from rotating, *regardless of the rotation direction*. On many machines, the miscellaneous function M05 must also be programmed *before* reversing the spindle rotation:

M03 (SPINDLE CW)

...

<... Machining at the current location ...> ...

M05 (SPINDLE STOP)

<... Usually, a tool change ... >

M04 (SPINDLE CCW)

...

<... Machining at the current location ...> ...

M05 function may also be required when changing gear ranges on CNC lathes (if available). A spindle stop programmed in a block containing an axis motion, will take place *after* the motion has been completed.

The last spindle control function is function M19, called the *spindle orientation*. Some control manufacturers call it the spindle key lock function. Regardless of the description, M19 function will cause the spindle to stop in an oriented position. This function is used mostly during machine setup, seldom in the program itself. Machine spindle must be oriented in two main situations:

Automatic tool change (ATC)

Tool shift during a boring operation

(G76 and G87 boring cycles only)

When *Automatic Tool Change* (ATC) function M06 is used in the program, there is no need to program spindle orientation for the vast majority of CNC machining centers. This orientation is built into the automatic tool changing sequence and guarantees the correct positioning of all cutting tool holders. Some programmers do like to program M19 with machine zero return motion for the tool change position, to save a second or two of the cycle time.

Spindle orientation is necessary for certain boring operations on milling systems. To exit a bored hole with a boring tool away from the finished cylindrical wall, the spindle must be stopped first, the tool cutting bit must be oriented, and then the tool can be retracted from the hole. A similar approach is used for back boring operations. However, these special cutting operations use fixed cycles in the program, where the spindle orientation is built in. For more details, *see pages 84 and 210*.

In conclusion, M19 miscellaneous function is rarely used in the program. It is available as a programming aid and to the CNC operator for setup work, using MDI operations.

Gear Range Selection

Virtually all programmable gear range selections apply to CNC lathes. On machining centers, spindle gear range is normally changed automatically. Most CNC lathes have two or more gear ranges available, some more powerful lathes are equipped with up to four selections. The basic programming rule is to select the gear range based on machining applications.

For example, most roughing operations require the *power* of the spindle more than the spindle *speed*. In this case, a low range is usually a better selection. For finishing work, a medium or high range is better, because high spindle rotation can be more beneficial to metal removing process.

Distribution of gear related miscellaneous functions depends entirely on the number of gears ranges the CNC lathe has available. Number of ranges is 1, 2, 3 or 4. The following table shows typical distribution of M-functions, but still check the actual commands in your machine tool manual.

Ranges	M function	Gear Range
1 available	N/A	None programmed
2 available	M41 M42	Low range High range
3 available	M41 M42 M43	Low range Medium range High range
4 available	M41 M42 M43 M44	Low range Medium range 1 Medium range 2 High range

Simple rule of thumb is that the higher the gear range, the more spindle speed is possible, and less spindle power is required. The opposite is also true. Normally, the spindle rotation does not have to be stopped to change a gear but consult the lathe manual anyway. If in doubt, stop the spindle first, change the gear range, then restart the spindle.

Machine Accessories

Most miscellaneous functions are used for some physical operation of machine tool accessories. From this group, the more common applications have been already covered, specifically coolant control and gear changes. The remaining M-functions in this group are described in detail elsewhere in this digital book, so only a short description is offered here.

The most notable of machine related M-functions are:

M function	Description	Type
M06	Automatic tool change (ATC)	M
M60	Automatic pallet change (APC)	M
M23 M24	Thread gradual pull-out <i>ON / OFF</i>	T
M98 M99	Subprogram <i>call</i> / Subprogram <i>end</i>	M/T

9. SEQUENCE BLOCK

Each line of a CNC program is called a *block*. In the terminology established earlier, a *block* was simply defined as a *single instruction processed by the CNC system*.

A sequence block, a program block - or simply a *block* is normally one handwritten line of the program listing, or a line typed in a text editor and terminated by the *Enter* key. This line can contain one or more program words - words that result in the definition of a *single instruction* to the CNC system. Such a program instruction typically contains combinations of preparatory commands, coordinate words, tool functions and commands, coolant function, speeds and feeds commands, position registration, offsets of different kinds, etc. In plain English, the contents of one block will be processed as a single unit *before* the control processes any following block of instruction. While a CNC program is being processed, the control system will evaluate individual instructions (blocks) *as one complete machine operation step*. Each part program consists of a series of blocks necessary to complete a certain machining process. The overall program length will always depend on the total number of blocks and their size.

A. BLOCK STRUCTURE

A single block allows as many program words as necessary. Some controls impose a limit on the number of characters in one block. There is only a theoretical maximum for Fanuc and similar controls, irrelevant in practice. The only restriction is that *two or more duplicated* words (functions or commands) cannot be used in the same block (with the exception of G codes). For example, only one miscellaneous M function or one coordinate word for the X axis in a single block are allowed. Most recent controls allow up to three M-functions in a block, providing they are not in conflict with each other. The order of individual words within a block follows a fairly free format - that means the required words may be in any order, providing the sequence block (N address) is written *first*. Although individual words are allowed in a block to be in any order, it is a standard practice to place words in a *logical order* within a block. It makes the CNC program easier to read and understand.

A typical program block structure is very dependent on the control system and CNC machine type. A typical block may contain several instructions, suggested in the order of entry. Not all program data are necessary to be specified in each block, only when required (to switch from one mode to another, for example). A block typically includes:

Block number N

Preparatory commands G

Auxiliary functions M

Axis motion commands X Y Z A B C U V W...

Words related to axes I J K R Q ... **Machine or tool function** S F T

The contents of a program block will vary between machine tools of different types, but logically, general rules will always be followed, regardless of the CNC system or the machine tool.

Building the Block Structure

Each block in a CNC program has to be developed with the same thought and care as any other important structure, for example a building, a car, or an aircraft. It starts with good planning. Decisions have to be made as to what will and what will not be part of the program block, similar to a building, car, aircraft or other structure. Also, important decisions have to be made as to what order of commands - or *instructions* - are going to be established within the block, and many other considerations.

The next few examples compare a typical structure of blocks for milling operations and blocks for turning operations. Each block is presented as a separate example.

Turning block examples:

N67 G00 G42 X50.0 Z2.5 T0202 M08 (EXAMPLE 1)

N23 G02 X75.0 Z-28.0 R5.0 F0.125 (EXAMPLE 2)

In the lathe examples (also metric), block N67 illustrates a rapid motion to an XZ position, as well as several other commands - the tool nose radius offset startup G42, activation of the tool offset (T0202), and the coolant ON function M08. The example in block N23 is a typical circular interpolation block with a feed rate of 0.125 mm/rev.

B. PROGRAM IDENTIFICATION

A CNC program can be identified by its number and - on some controls - also by its name. Identification by the program number is necessary in order to store more than one program in the CNC memory. Program name or a brief description, if supported, can be viewed on the control screen display.

Program Number

The first block used in any part program is commonly a program number, and must be specified in the program, if the control system requires it. Two addresses are available for the program number - the capital letter O for EIA format, and the older colon [:] for ASCII (ISO) format. In memory operation, the control system always displays program number with the letter O. Program number block is *not always* necessary to include in the CNC program and often it is better to let the CNC operator make the selection.

If a program does use program numbers, they must be specified within a certain range. Programs for older Fanuc controls must be within the range of O1 - O9999, program number zero (O0 or O0000) is not allowed. Newer controls allow 5-digit program numbers, in range of O1 - O99999. Neither a decimal point nor a negative sign is not allowed in any program number. Leading zeros suppression is normal - for example, O1, O01, O001, O0001, and O00001 are all legitimate entries, in this case for a program number *one*.

Program Name

On the latest Fanuc control systems, the program name can be included *in addition* to the program number, *not instead* of it. Program name (or a brief program description) can be up to sixteen characters long (spaces and symbols *are* counted). The program name must be *on the same line* (in the same block) as the program number:

O1001 (DWG. A-124D IT.2)

There is a distinct advantage of displaying the program number along with its description - it makes the directory listing more descriptive and useful.

Watch carefully *where* the description is written. If the program name is longer than sixteen characters allowed, no error is generated, but only the first sixteen characters will be displayed. Make sure to avoid program names that can be ambiguous when displayed. Consider these two program names, they both appear to be correct:

O1005 (LOWER SUPPORT ARM - OP 1)

O1006 (LOWER SUPPORT ARM - OP 2)

Since the control screen display can only show the *first sixteen* characters of program name, the program names will be ambiguous when displayed:

O1005 (LOWER SUPPORT AR)

O1006 (LOWER SUPPORT AR)

To eliminate this problem, use an abbreviated description that falls *within* the sixteen characters

and still contains all significant information - for example:

O1005 (LWR SUPP ARM OP1)

O1006 (LWR SUPP ARM OP2)

If a detailed description is required, it has to be divided over one or more comment blocks:

O1005 (LWR SUPP ARM OP1)

(OPERATION 1 - ROUGHING)

Specified comments in the block or blocks following the program number will *not* appear on the directory screen listing, but still will be a useful aid to the CNC operator. They will be displayed during the program processing and, of course, listed in a hard copy printout.

Program *names* should be short and descriptive - their purpose is to assist the CNC operator when searching for programs stored in the control memory. Any suitable information could include a drawing number or a part number, shortened part name, operation, etc. Descriptions not suitable in such blocks include machine model, control system, programmer's name, dates and times, company or customer's name and similar data - they can be part of program header, described earlier - *see page 50*.

On most controls, when loading part program into the memory, CNC operator must specify the program number, regardless of what the actual number is in the program. *Program number entered at the control always supersedes the programmer specified number*. It can be a number that just happens to be available (unused), or it can be a number that has a unique meaning, for example, a unique group (for example, all programs that begin with the O10xx belong to the group associated with a single customer). Subprograms are different - they must always be stored using the number specified by the CNC programmer. Innovative use of program numbers may also serve to keep track of programs developed for each machine or part.

C. SEQUENCE NUMBERS

Individual program sequence blocks can be referenced with a number for easier orientation within the program. The program address for a block number is the letter N, followed by up to five digits - from sequence 1 to sequence 9999 or 99999, depending on the control system. The block number range will typically be N1 to N9999 for older controls and N1 to N99999 for newer controls. Some rather old controls accept block number in the three-digit range only, N1 to N999.

The N address must always be the *first word* in the block. For an easier orientation in programs that use subprograms, there should be no duplication of block numbers between both types. For example, a main program starting with N1 and a subprogram also starting with N1 may cause a confusing situation. Technically, there is nothing wrong with such a designation. Refer to *page 387* for suggestions on block numbering in subprograms.

N0 and N negative or with a decimal point are not allowed

Sequence Number Command

In the table below, the first column represents sequence numbers used normally, the second column shows the sequence numbers required in a format acceptable to the machine control system, as applied to a CNC program:

Increment	First block number
1	N1
2	N2

5	N5
10	N10
50	N50
100	N100

Using sequence numbers (block numbers) in a CNC program offers two advantages and at least one possible disadvantage:

On the positive side, block numbers will make any program search much simplified during editing or tool repetition at the machine. They also make the program much easier to read when displayed during processing or read on paper. They mean both *the programmer* as well as *the operator* benefit

On the negative side, block numbers will *reduce* the available control system memory storage. That means a fewer number of programs can be stored in the memory, and long programs may not fit in their entirety

Sequence Block Format

Block number program input format notation, using the N address, is N5 for the more advanced controls and N4 or even N3 for older controls. Block number N0 is not allowed, neither is a minus sign, a fractional number, or the use of a decimal point. Minimum block increment number must always be an integer - smallest integer allowed is one (N1, N2, N3, N4, N5, etc.). A larger increment is allowed, and its selection depends on your personal programming style or standards established within company.

Typical sequence block increments other than one are:

Increment	Program example
2	N2, N4, N6, N8, ...
5	N5, N10, N15, N20, ...
10	N10, N20, N30, N40, ...
100	N100, N200, N300, N400, ...

Some programmers like to use increments of 5, 10 or even more. There is nothing wrong with this method, but the CNC program will become unnecessarily too long, too soon, and possibly difficult to manage. Block numbers do occupy memory space.

In all cases of block increments other than one, the intent is the same - to allow for additional blocks to be filled-in between existing blocks, if such a need arrives. The need may arise while proving or optimizing the program during test run, where an addition to the existing program will be required. Although all new blocks (the ones inserted) will *not* be in the order of an equal increment, at least they will be numerically ascending. For example, a face cut on a lathe with a single cut (*Example A*) was modified by the machine operator for two cuts (*Example B*):

Example A - single face cut:

N40 G00 G41 X85.0 Z0 T0303 M08

N50 G01 X-1.8 F0.2 N60 G00 W3.0 M09

N70 G40 X85.0

...

Example B - *Example A* modified for two face cuts:

N40 G00 G41 X85.0 Z1.5 T0303 M08

N50 G01 X-1.8 F0.2

N60 G00 W3.0

N61 X85.0

N62 Z0

N63 G01 X-1.8 N64 G00 W3.0 M09

N70 G40 X85.0 ...

Note the change in block N40 and added blocks N61 to N64. Preference in this digital book is to program in increments of one and if an addition is needed, the added blocks will have *no block numbers at all* (check if the control system allows block numbers to be omitted, most do).

Example C - single face cut:

N40 G00 G41 X85.0 Z0 T0303 M08

N41 G01 X-1.8 F0.2

N42 G00 W3.0

N43 G40 X85.0

Example D - *Example C* modified for two face cuts:

N40 G00 G41 X85.0 Z1.5 T0303 M08

N41 G01 X-1.8 F0.2

N42 G00 W3.0

X85.0

Z0

G01 X-1.8

G00 W3.0

N43 G40 X85.0

Note that the program is a little smaller and the additional blocks are quite visual and noticeable when printed or displayed on the screen.

Leading zeros may (and should) be omitted in the block number - for example, N00008 should be written as N8. Omitting leading zeros will significantly reduce the overall program length. Trailing zeros must always be written, to distinguish for such similarities as N08 and N80.

Use of block numbers in a part program is optional, as shown in the earlier example. A program containing block numbers is easier to read. For a CNC operator, search and edit functions in program editing are also easier. *Note* some programming applications depend on the block numbers, for example, lathe multiple repetitive cycles G70, G71, G72, G73. In this case, *at least* the significant blocks have to be numbered (*see page 324*).

Numbering Increment

Fanuc block numbers in a program can be in any physical order - ascending, descending or mixed - they can also be duplicated or missing altogether. Some programming practices are established as preferable, because they are logical and make sense. Having a mixed order of sequence numbers in a program serves no useful purpose and neither do duplicate sequence numbers. If a program contains duplicate block numbers and a block number search is initiated,

the control system will only search for the *first* occurrence of the particular block number, which may or may not be the one expected. Any further search will have to be repeated from the string found last. The reason for such a generous latitude in sequence block numbering is to offer flexibility to CNC operator *after* the program has been done and loaded into the control.

Block sequence number does not affect the order of program processing, regardless of its increment. Even if the blocks are numbered in a descending or mixed order, the part program will always be processed sequentially, on the basis of the block *contents*, not its number. The increment of 5 or 10 is the most practical, since it allows for insertion of up to 4 to 9 blocks respectively between any two original blocks. That should be more than sufficient for the majority of program modifications. Normally, this digital book uses block increment of one (N1, N2, N3, ...).

For those CNC programmers who use a computer-based programming system, just a few words relating to the programming of sequence numbers. Although the computer programming system allows the start number of the block and its increment to almost any combination, adhere to *the start and increment numbers of one* (N1, N2, N3, ...). The purpose of a computer-based programming is to keep an accurate database of the part geometry *and* the cutting tool path. If the CNC program is modified manually, the part computer database is not accurate anymore. Any CNC program change should always be reflected in the *source* of the program, as well as its result - never in the result alone.

Long Programs and Block Numbers

Long programs are always difficult to load into a CNC memory with limited capacity. In such cases, the program length may be shortened by omitting block numbers altogether or - even better - by programming them only where really needed - for all *significant blocks*. Significant blocks are those that have to be numbered for purposes of program search, tool repetition, or other procedure that depends on program numbers, such as machining cycles or tool changes. In these cases, select increments of two or five, for the operator's convenience. Even limited use of sequence numbers will increase the program length, but for a justifiable reason.

If *all* sequence block numbers have been omitted in the program, the search on the machine control will become rather difficult. CNC operator will have no other option but to search for the next occurrence of a particular address within a block, such as X, Y, Z, etc., rather than a sequence block number. This method of search may unnecessarily prolong the search time.

D. END OF BLOCK CHARACTER

Because of the control system specifications, individual sequence blocks must be separated by a special character, known as the *end-of-block* character or by its abbreviation EOB or E-O-B. On most computer systems, the EOB character is generated by the *Enter* key on the keyboard. When a program is input to the control by MDI (Manual Data Input), EOB character key on the control panel terminates the block. The end-of-block symbol on Fanuc controls appears as a semicolon [;].

The displayed semicolon symbol is only a *graphic representation* of the end-of-block character and is never entered literally in the CNC program. Under no circumstances it should be included in the program itself. Some older control systems have an asterisk [*] as the display symbol for the end-of-block, rather than the semicolon [;]. Many controls use other symbols, that also represent the end of block, for example, some use the dollar sign [\$]. In any case, remember such symbol is only a *representation* of the end-of-block character, *not* its actual character.

E. STARTUP BLOCK OR SAFE BLOCK

A *startup block* (sometimes called a *safe block* or a *status block*) is a special sequence block. It contains one or more modal words (usually preparatory commands of several G groups) that preset the control system into a desired *initial* or *default* state. This block is placed at the beginning of each program or even at the beginning of each tool and it is the *first block* processed during a

repetition of a program (or a tool within a program). In CNC program, the startup block usually precedes any motion block or axis setting block, as well as the tool change or tool index block. This is the block to be searched for, if the program or the desired cutting tool is to be repeated during machine operation. Such a block will be slightly different for the milling and turning systems, due to the unique requirements of each control system.

Earlier in this digital book (see page 23) one topic covered state of the control system when the main power has been turned on - which activates the *system default* settings. A CNC programmer should never count on these default settings or conditions, since they can be easily changed by the machine tool operator, without programmer's knowledge. If such a change does happen, the programmed settings will *not* correspond to those suggested by the machine tool manufacturer or the engineers who designed the control system.

A careful CNC programmer should always adopt the attitude of a *safe* programming approach and will not leave anything to chance. Programmer will try to preset all required conditions under the program control, rather than counting on the defaults of a CNC system. Such an approach is not only much safer, but it will also result in programs that are easy to use during setup, tool path verification and tool repetition due to tool breakage, dimensional adjustments, etc. It is also very beneficial to the CNC machine operators, particularly to those with limited experience.

In all applications listed, startup block will not increase the machining cycle time in any way. Another benefit of a startup block is that the program is more transportable from one machine tool to another, since it does not count on the default settings of a particular machine-control combination.

The name *safe block* - which is another name used for a startup block - does not become *safe* on its own - it must be *made* safe. Regardless of the name, this block should contain all control settings for the program or cutting tool that start the program in a 'clean' state. The most common entries that set this initial status are the dimensioning system (metric/imperial and absolute/incremental), cancellation of any active cycle, cancellation of active cutter radius offset mode, plane selection for milling, feed rate default selection for lathes, etc. Examples presented here show some startup blocks for both milling and turning controls.

At the program beginning (for milling), a startup block may be programmed with the following contents:

N1 G00 G17 G21 G40 G54 G64 G80 G90 G98

N1 block is the first sequence number, G00 selects the rapid mode, G17 establishes the XY plane selection, G21 selects metric units, G40 cancels any active cutter radius offset, G64 sets a continuous cutting mode, G80 cancels any active fixed cycle, G90 selects the absolute mode, and G98 will retract to the initial level in a fixed cycle. These conditions apply only when the startup block is processed as the *first major block* in CNC program - any subsequent program changes will become effective only with the block in which the change is applied. For example, if G01 command is effective by default, any subsequent usage of G00, G02, or G03 will cancel the G01 command.

At the beginning of a CNC lathe program, the startup block may contain these G codes:

N1 G21 G00 G40 G99

N1 is the first block number, G21 selects metric units, G00 selects the rapid mode, G40 cancels any active tool nose radius offset, and the G99 selects feed rate per revolution mode. Reference to absolute or incremental system is usually not required, since lathe controls use addresses X and Z for *absolute* dimensioning and addresses U and W for *incremental* dimensioning. For lathe controls that do not support the U and W addresses, the standard G91 code is used for incremental values in X and Z axes. As in the milling example, any of the words programmed in the safe block can be overridden by subsequent change of G-codes.

Some controls systems do not allow certain G-codes on the same line. For example, G20 or G21 may *not* be programmed with other G-codes. If you are not sure, place these G-codes in separate blocks. Instead of ...

N1 G21 G17 G40 G49 G80

... two or more blocks can be safely used:

N1 G21

N2 G17 G40 G49 G80

F. PROGRAM COMMENTS

One or more program comments and messages can be included within the program body as separate blocks, or as parts of an existing block, mostly in cases when the message is short. In either case, the message must be enclosed in parenthesis (for ASCII/ISO format):

Example A:

N330 M00 (REVERSE PART) Example B:

N330 M00 (REVERSE PART / CHECK TOOL) Example C:

N330 M00

(REVERSE PART / CHECK TOOL)

The purpose of a message or comment in the program is to inform the machine operator of a specific task that must be performed *every time* program reaches the stage of processing where such message appears. Comments are also useful for understanding the program at a later date and can be used for documenting the program.

Typical messages and comments relate to information about setup changes, chip removal from a hole, dimensional check, cutting tool condition check and many others. A message or a comment block should be included only if the required task is *not clear* from the program itself - no need to describe what happens in each block. Messages and comments should be brief and focused, as they occupy a memory space in the CNC memory.

From practical perspective, short series of messages and comment blocks can be provided at the beginning of each program, to list all significant drawing information and cutting tools required for the job. This subject has been covered on *page 50* - here is just a reminder:

O1001 (SHAFT - DWG B451)

(SHAFT TOOLING - OP 1 - 3 JAW CHUCK)

(T01 - ROUGH TOOL - 1/32R - 80 DEG)

(T02 - FINISH TOOL - 1/32R - 55 DEG)

(T03 - OD GROOVING TOOL - 0.125 WIDE)

(T04 - OD THREADING TOOL - 60 DEG)

N1 G20 G99 N2 ...

If the available memory space of CNC unit is limited, using comment blocks in this manner may prove impractical. It will be better if the messages and comments are listed in proper setup and tooling sheets, with all required details.

G. CONFLICTING WORDS IN A BLOCK

All instructions in a program block must be logical and reasonable - not impossible. For instance, the first block of a program contains the following words:

N1 G20 G21 G17

What the block contains is simply not logically possible. It instructs the control to:

'Set the imperial system of dimensions, also set the metric system of dimensions and set the XY plane'.

Definitely not possible, but also not realistic - what will actually happen and how does the control interpret such a statement? The XY-plane is all right, but what about the selection of dimensions? Obviously, both selections are not possible - the block contains *conflicting* words - opposite dimensional units. Some controls may give an error message, Fanuc systems will not. What will happen? The control unit will evaluate the sequence block and check for any words within the same group. Distribution of G-code groups have been described in the section dealing with preparatory commands - G codes (*see page 55*).

If the computer system finds two or more words that belong to the same group, it will not return an error message, it will automatically activate the *last* word of the group. In the example of conflicting dimensional selection, it will be the preparatory function G21 - selection of metric dimensions - that becomes active. That may or may not be the selection required. Rather than counting on some kind of elusive luck, make sure there are no conflicting words in any program block.

In the example illustrating the metric and imperial selection, the preparatory command G was used. What would happen if, for example, the address X was used? Consider the following example:

N120 G01 X11.774 X10.994 Y7.056 F15.0

There are two X addresses in the same block. Control system *will not* accept the second X-value, but it will issue an alarm (error). Why? Because there is a great difference between programming rules for a G-code as such and for coordinate system words. Fanuc controls allow to place as many G-codes in the same block as needed, providing they are not in conflict with each other. But the same control system will not allow to program more than one coordinate word of the same address for each sequence block. Some other rules may also apply. For example, the words in a block may be programmed in any order, providing the block N-address is the first one listed. For example, the following block is legal (but very nontraditional in its order):

N340 Z-0.75 Y11.56 F10.0 X6.845 G01

As a matter of good programming practices, be sure to write the entries for each sequence block in a logical order. Block number must be the first word and is usually followed by G-code(-s), primary axes in their alphabetical order X., Y., Z.), auxiliary axes or modifiers (I., J., K.), miscellaneous functions and words, and the feed rate word as the last item. Select only those words needed for the individual block:

N340 G01 X6.845 Y11.56 Z-0.75 F10.0

Two other possibilities exist that may require a special attention in programming approach. For example, how will the following block be interpreted?

N150 G01 G90 X5.5 G91 Y7.7 F12.0

N151 (INCREMENTAL MODE IN EFFECT)

There is an apparent conflict between the absolute and incremental modes. Most Fanuc controls will process this block exactly the way it is written (check first). The X-axis target position will be reached in absolute mode, but the Y-axis will be an incremental distance, measured from tool current position. It may not be a typical approach, but it offers advantages in some cases. Remember - the sequence block following block N150 will be in *incremental* mode, since G91 is specified *after* the G90 command!

Another programming application to watch for, is in a block programmed while circular interpolation mode is active. Section dealing with this subject starts on *page 243* and specifies that an arc or a circle can be programmed either with arc vectors I, J and K (depending on whether a milling or a turning control system is used). It also specifies that a direct radius input, using the address R, can be used. Both of the following examples are correct, resulting in a 90 arc with a 1.5-inch radius:

With I and J arc vectors:

N21 G01 X15.35 Y11.348

N22 G02 X16.85 Y12.848 I1.5 J0 N23 G01 ...

With the direct radius R-address:

N21 G01 X15.35 Y11.348

N22 G02 X16.85 Y12.848 R1.5 N23 G01 ...

Now, consider how the control system will process block N22, if it contains *both*, the I and J vectors as well as the radius input:

N22 G02 X16.85 Y12.848 I1.5 J0 R1.5

or

N22 G02 X16.85 Y12.848 R1.5 I1.5 J0

The answer may be surprising - in *both cases*, the control will ignore the I and J values and *will only process the amount of radius R*. Actual order of address definition is irrelevant in this special case. The address R has a higher control priority than I and J addresses, if programmed in the same block. All examples assume that the control system supports R-radius input.

H. MODAL PROGRAMMING VALUES

Many program words are modal. The word *modal* is based on the word '*mode*' and means that a specific command remains in this mode after it has been used in the program once. It can only be canceled by another modal command of the same group. Without this feature, a program using linear interpolation in absolute mode with a feed rate of 18.0 in/min, would contain absolute command G90, linear motion command G01 and feed rate F18.0 *in every block*. With modal values, the programming output is much shorter. Virtually all controls accept modal commands. The following two examples illustrate the differences:

Example A - *without* modal values:

N12 G90 G01 X1.5 Y3.4 F18.0

N13 G90 G01 X5.0 Y3.4 F18.0

N14 G90 G01 X5.0 Y6.5 F18.0

N15 G90 G01 X1.5 Y6.5 F18.0

N16 G90 G01 X1.5 Y3.4 F18.0 N17 G90 G00 X1.5 Y3.4 Z1.0

Example B - *with* modal values:

N12 G90 G01 X1.5 Y3.4 F18.0

N13 X5.0

N14 Y6.5

N15 X1.5

N16 Y3.4

N17 G00 Z1.0

Both examples will produce identical results. Compare each block of *Example A* with the corresponding block of *Example B*. Observe that the modal commands are *not* necessary to be repeated in CNC program. In fact, in everyday programming, many program commands used are modal. The exceptions are those program instructions, whose functionality starts and ends in the same block (for example dwell, machine zero return, certain machining instructions, such as tool change, indexing table, etc.). M-functions behave in a similar fashion. For example, if the program contains a machine zero return in two consecutive blocks (usually for safety reasons), it may look like this:

N83 G28 Z1.0 M09

N84 G28 X5.375 Y4.0 M05

G28 cannot be removed from block N84, because the G28 command is *not* modal and must be repeated.

I. EXECUTION PRIORITY

There are some special cases, mentioned earlier, where the *order* of commands in a block determines the priority in which the commands are executed. To complete the subject of a sequence block, let's look at another situation. Here are two unrelated blocks used as examples:

N410 G00 X22.0 Y34.6 S850 M03 and

N560 G00 Z5.0 M05

In the block N410, rapid motion is programmed together with two spindle commands. What will *actually happen* during program execution? It is very important to know *when* the spindle will be activated in relationship to the cutting tool motion. On Fanuc and many other controls, the spindle function will take effect *simultaneously* with the tool motion.

In block N560, Z-axis (Z5.0) motion is programmed, this time together with the spindle stop function (M05). Here, the result will be different. Machine spindle will be stopped only when the motion is *one hundred percent completed*. For more details relating to M-functions used with tool motion, see tables on *page 60*.

Similar situations exist with a number of miscellaneous functions (M-functions), and any programmer should find out exactly how a particular machine and control system handle a motion combined with an M-function address in the same block. Here is a refresher in the form of a list of the most common results:

Functions that will be executed *simultaneously* with tool motion are:

M03	M04	M07	M08

Functions that will be executed *after* the tool motion has been completed are:

M00	M01	M05	M09	M98

Be careful here - if in doubt, program it safe. Some miscellaneous functions require an *additional* condition, such as another command or function to be active. For example, M03 and M04 will only work if the spindle function S is in effect (spindle is rotating). Other miscellaneous functions should be programmed in separate blocks, many of them for logical or safety reasons:

M02	M06	M10	M11	M19	M30	M60	M99

Functions indicating the end of program or subprogram (M02, M30, M99) should stand on their own and not be combined with other commands in the same block, except in special cases. Functions relating to machine mechanical activity (M06, M10, M11, M19, M60) *should* be

programmed without any motion in effect, for safety. In case of M19 (spindle orientation), the spindle rotation must be stopped first, otherwise machine may get damaged. Not all M-functions are listed in the above examples, but they should provide a good understanding of how they may work, when programmed together with a motion. The chapter describing the miscellaneous functions also covers the duration of typical functions within a program block.

It never hurts to play it safe and always program these possible troublemakers in a sequence block containing no tool motion. For mechanical M-functions, make sure the program is structured in such a way that it provides safe working conditions - these functions are oriented mainly towards the machine setup.

10. INPUT OF DIMENSIONS

Addresses in a CNC program that relate to the tool position at a given moment are called *the coordinate words*. Coordinate words always have a dimensional value, using currently selected units - *metric* or *imperial*. Typical coordinate words are X, Y, Z, I, J, K, R, etc. They are the basis of all dimensions in CNC programs. Tens, hundreds, even thousands of calculations may have to be made to develop a program that will do what it is intended to do - to *accurately machine a complete part*.

Dimensional entries in a program assume two attributes:

Dimensional units ... Imperial or Metric

Dimensional references ... Absolute or Incremental

Unit of dimensions in a program can be one of two kinds - *metric* or *imperial*. The reference of dimensions can be either *absolute* or *incremental*.

Fractional values, for example 1/8, are not allowed in a CNC program and have to be converted to their decimal equivalent. In metric format, *millimeters* and *meters* are used as units, in imperial format it is *inches* and *feet* that form the basis of units. Regardless of format selected, the number of programmed decimal places can be controlled, suppression of leading and trailing zeros can be set, and decimal point can be programmed or omitted, as is applicable to a particular CNC system.

A. IMPERIAL AND METRIC UNITS

Drawing dimensions can be used in a program in either *metric* or *imperial* units. This digital book uses the combined examples of both the imperial (English) system, still common in the USA, to some extent in Canada, and one or two other countries. The metric system is common in the rest of the world. With economy reaching global markets, it is important to understand both systems. The use of metric system is on the increase even in countries that still use imperial units of measurement, mainly the United States.

Machines that come equipped with Fanuc controls can be programmed in either mode. Initial CNC system selection (known as the *default* condition) is controlled by a parameter setting of the control system but can be overridden by a preparatory command written in the part program. Default conditions are usually set by the machine tool manufacturer or distributor. Often, it is based on engineering design decisions, as well as the demands of their customers.

During program development, it is imperative to consider the impact of control system default conditions on program execution. Default conditions come into effect the moment a CNC machine has been *turned on*. Once a command is issued, in MDI mode or in a program, many default values may be overwritten and will remain changed from that point on. Dimensional unit selection in the CNC program will change the *default* value - that is the internal control setting. In other words, if metric unit selection is made, the control system will remain in that mode until an imperial selection command is entered. That can be done either through the MDI mode, a program block, or a system parameter. This applies even for situations when the power has been turned *off* and then *on* again!

To select a specific dimensional input, regardless of default conditions, a preparatory G command is required at the beginning of CNC program:

G20	Selects imperial units (inches and feet)
G21	Selects metric units (millimeters and meters)

Without specifying the preparatory command in the program, control system will default to the status of current parameter setting. Both preparatory command selections are *modal*, which means the selected G code remains active until the opposite G code is programmed - so metric units will

be active until the imperial system replaces it and vice versa.

This reality may suggest a certain freedom of switching between the two units anywhere in the program, almost at random and indiscriminately. *This is not true.* All controls, including Fanuc, are based on the *metric* system, partially because of the Japanese influence, but mainly because the metric system is more accurate. Any ‘switching’ by use of the G20 or G21 command does not necessarily produce any real conversion of one unit into the other, but merely *shifts* the decimal point, not the actual digits. At best, only some conversions take place, not all. For example, G20 or G21 selection will convert one measuring unit to another on *some - but not all* - offset screens. Many controls will convert all settings, but even then it is *not* recommended to mix the two-unit systems in the same program. Following two examples will illustrate the *incorrect* result of changing G21 to G20 and G20 to G21 within the same program. Read the comments attached to each block - you may find a few surprises:

Example 1 - from metric to imperial units:

G21 *Initial units' election(metric)*

G00 X60.0 *X value is accepted as 60mm*

G20 *Previous value will change into 6.0 inches*

(Real translation is 60mm=2.3622047inches) Example 2 - from imperial to metric units:

G20 *Initial units' election(imperial)*

G00 X6.0 *X value is accepted as 6.0 inches*

G21 *Previous value will change into 60 mm*

(Real translation is 6.0 inches=152.4mm)

Both examples illustrate the possible problem caused by switching between two dimensional units in the same program. For this reason, always use only one-dimensional unit in a part program. If the program calls a subprogram, the rule extends to subprograms as well:

Never mix metric and imperial units in the same program

In fact, it is ill-advised to mix them, even if the results for the control system are predictable and full conversion is available. The selection of dimensional system will make a great difference how some control functions will work. The following functions *will be* affected by the change from one system of units to the other:

Dimensional words (X, Y, Z axes, I, J, K modifiers, etc.)

Constant Surface Speed (CSS)

(Known as *cutting* speed, used by CNC lathes)

Feed rate function (F-address)

Offset values (H and D offsets for milling and tool preset values for lathes)

Screen position display (number of decimal places)

Manual Pulse Generator - the HANDLE (value of divisions)

Some control system parameters

The initial selection of dimensional units can also be done by a system parameter setting. Control status when the power has been turned on is the same as it was at the time of last power shut off. If neither G20 nor G21 is programmed, the control accepts dimensional units selected by a parameter setting. If G20 or G21 is included in the program, *the program command will always take a priority* over any control system parameter setting. Programmer makes all decisions - the control system is only interpreting them, but it does not mean it is always ‘right’.

Dimensional units setting should always be in a separate block, *before* any axis motion, offset selection, or even setting of a coordinate system (G92, G50 and G54 to G59) - in other words, it should be in the first program block. Failure to follow this rule may produce incorrect results, particularly when frequently changing units for different jobs.

Comparable Unit Values

There are many units available in both metric and imperial systems. In CNC programming, only a very small portion of them is used. Metric units are based on a *millimeter* or a *meter*, depending on application. Imperial units are based on *inches* and *feet*, again, depending on application. Common abbreviations for different units are:

Millimeter	mm
Meter	m
Inch	in
Foot	ft

Many programming terms use these abbreviations. The next table shows comparable terms between the two-dimensional systems (older terms are in parentheses):

Metric	Imperial
m/min (also MPM)	ft/min (also FPM or SFPM)
mm/min	in/min (also IPM or ipm)
mm/rev	in/rev (also IPR or ipr)
mm/tooth	in/tooth (also IPT or ipt)
kW	HP

B. ABSOLUTE AND INCREMENTAL MODES

A dimension in either input units must have *a specified point of reference*. For example, if X35.0 appears in the program and currently selected units are millimeters, the statement does not indicate *where* the dimension of 35 mm has its origin. Control system needs more information to interpret dimensional values correctly, as intended.

There are two types of references in CNC programming:

Reference to a *common point* on the part

... known as the origin for ABSOLUTE input

Reference to a *previous point* on the part

known as the last tool position for INCREMENTAL input

In the example, X35.0 dimension (and any other as well) can be measured from a selected fixed point of the part, called *origin*, or *program zero*, or *program reference point* - all these terms have the same meaning. The same value of X35.0 can also be measured from the *previous* position, which is always the *last* tool position. This position then becomes the *current* position for the *next* tool motion. Control system cannot distinguish one of the two possibilities from the X35.0 statement alone, and some other description must be added to the program.

All dimensions in a CNC program measured from the *common point* (origin) are *absolute*

dimensions, as illustrated in *Figure 11-1*, and all dimensions in a program measured from the *current position* (last point) are *incremental dimensions*, as illustrated in *Figure 11-2*.

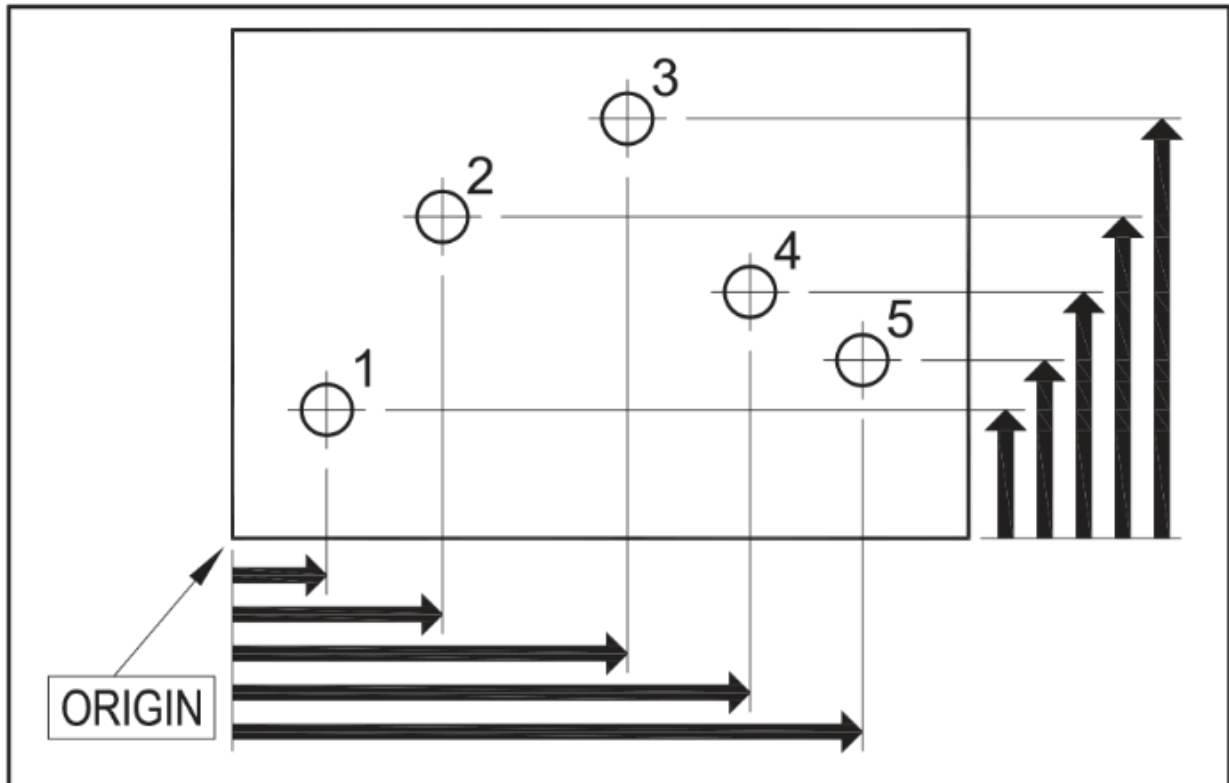


Figure 11-1

Absolute dimensioning - measured from part origin

G90 command will be used in the program

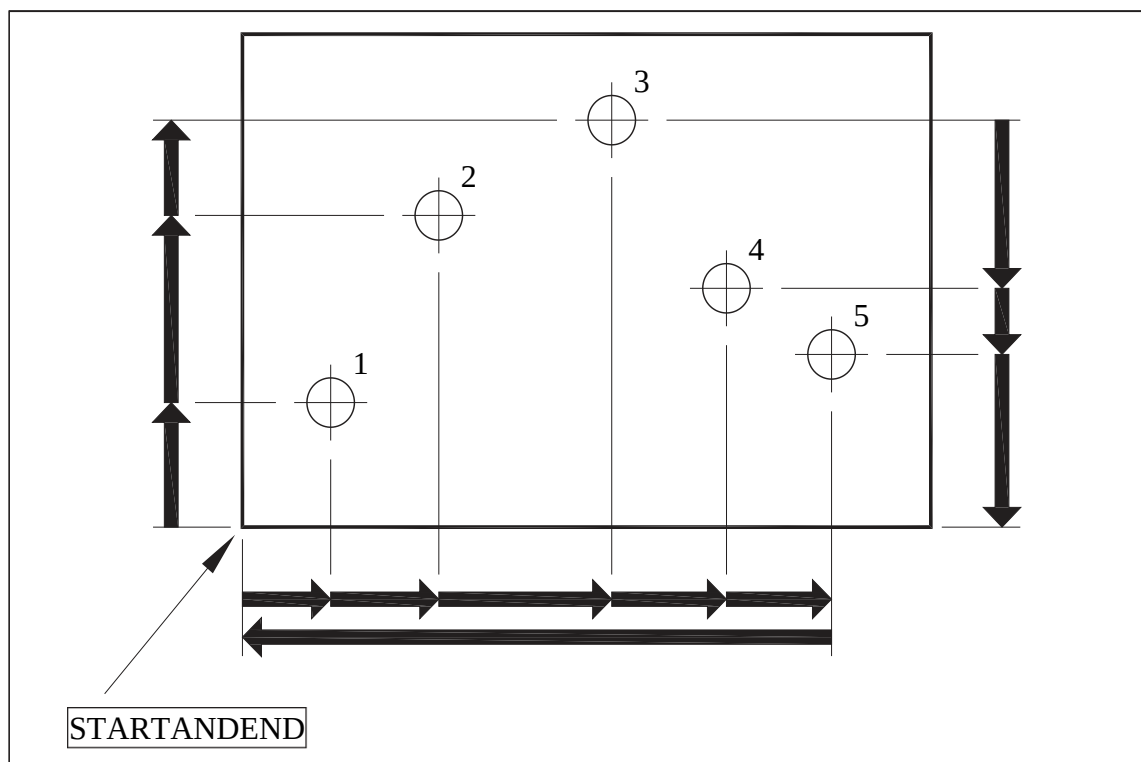


Figure 11-2

Incremental dimensioning - measured from the current tool location

G91 command will be used in the program

Absolute dimensions in a program represent target locations of the cutting tool from origin

Incremental dimensions in a program represent the actual amount and direction of the cutting tool motion from the current location

Since the dimensional address X, written as X35.0, is programmed the same way for *either* point of reference, some additional means must be available to the programmer. Without them, the control system would use default system parameter setting, not always reflecting the programmer's intentions. Selection of the dimensioning mode is controlled by two modal G commands.

Preparatory Commands G90 and G91

There are two preparatory commands available for the input of dimensional values - G90 and G91 - to distinguish between two available modes:

G90	Absolute mode of dimensioning
G91	Incremental mode of dimensioning

Both commands are modal, therefore they will cancel each other. The control system uses an initial default setting when powered on, which is usually the *incremental mode*. This setting can be changed by a system parameter that presets the computer at the power startup or a reset. For individual CNC programs, system setting can be controlled by including the proper preparatory command in the program, using either one of two available commands - G90 or G91.

It is a good programming practice to always include any required setting in the CNC program, and not to count on any default control system settings. It may come somewhat as a surprise that the common default control setting is the *incremental mode*, rather than the absolute mode. After all, absolute programming has a lot more advantages than incremental programming and is far more popular. In addition, even if incremental programming is used frequently, every program still starts up in the absolute mode. So, the question is *why* the incremental default? There is a good reason for it - as in many cases of defaults - *machining safety*. Follow this reasoning:

Consider a typical start of a new program loaded into the machine control unit. Control had just been turned on, part is safely mounted, current cutting tool is at the home position, offsets are set, and the program is ready to start. Such a program is most likely written in more practical absolute mode. Everything seems fine, except that the absolute G90 command is missing in the program. *What will happen at the machine?* Think before an answer and think logically.

When the first tool motion command is processed, the chances are that its target values will be positive or have small negative values. Because any dimensional input mode is missing in the program, control system '*assumes*' the mode as being incremental, which is the normal default value stored in a system parameter. The active tool motion, generally in X and Y axes only, will take place to either the overtravel area, in the case of positive target values, or by a small amount, in case of negative target values. In either case, the chances are that no damage will be done to the machine or the part. Of course, there is no guarantee, so always program with safety in mind.

G91 is the standard default mode for input of dimensions

Absolute Data Input - G90

In absolute programming mode, all dimensions are measured from the *point of origin*. Origin is the *program reference point*, also known as *program zero*. Actual motion of the machine is the real difference between the current absolute tool position and its previous absolute position. Algebraic signs [+] plus or [-] minus refer to the quadrant of rectangular coordinates, *not* the direction of motion - *see page 16 for details*. Positive sign does not have to be written for any address. All zero values, such as X0, Y0 or Z0 refer to the tool position at program reference point, *not* to the tool motion itself. Zero value of any axis must be written when such tool position is

required.

The preparatory command G90, selected for absolute mode setting, remains modal until the incremental command G91 is programmed. In absolute mode, there will be no motion for any axis that is omitted in the program.

Probably the main advantage of absolute programming is ease of modification by either the programmer or CNC operator. A change of one dimension does not affect any other dimension in the program.

For CNC lathes with Fanuc controls, the common representation of the absolute mode is the axis designation as X and Z, *without* the G90 command. If available, rotation axis uses the address C in absolute mode. Some lathes may use the G90 command, but not those with Fanuc controls.

Incremental Data Input - G91

In incremental mode of programming, also called a *relative* mode, all program dimensions are measured as departure distances into a specified direction (equivalent to '*distance-to-go*' on the control system). Actual motion of the machine is the specified amount along each axis, with the direction indicated as positive or negative.

The signs + or - specify *direction* of the tool motion, *not* the quadrant of rectangular coordinates. Plus-sign for positive values does not have to be written, but minus sign must be used. All zero input values, such as X0, Y0 or Z0 mean there will be *no tool motion along that axis*, and *do not* have to be written at all. If a zero-axis value is programmed in incremental mode, it will be ignored. The preparatory command for incremental mode is G91 and remains modal until the absolute command G90 is programmed. There will be no motion of any axis omitted in the program block.

The main advantage of incremental programs is their portability between individual sections of a program. An incremental program can be called at different locations of the part, even in different programs. It is mostly used when developing subprograms or repeating an equal distance.

For Fanuc controlled CNC lathes, the most common representation of the incremental mode is the axis designation as U and W (for X and Z respectively), *without* the G91 command. If available, rotation axis uses the address H in incremental mode. Some lathes may use the G91 command, but not those with Fanuc controls. **Combinations in a Single Block**

On most Fanuc controls, absolute and incremental modes can be combined in a single program block for special programming purposes. This may sound rather unusual, but there are significant benefits in this advanced application. Normally, a block is in one mode only - *either* in absolute mode *or* in incremental mode. On many controls, for any change over to the opposite mode, motion command must be programmed in a separate block. Such controls, for example, do not allow programming incremental motion along one axis and absolute motion along the other axis *within the same block*.

Most Fanuc control systems *do* allow programming both modes *within the same block*. Just specify G90/G91 preparatory command *before* each significant address.

For lathe work, where G90 and G91 are not used, the changeover is between X and U axes and Z and W axes. The X and Z contain absolute values, U and W are incremental values. Both types can easily be written within the same block. Here are two typical examples:

Turning example (G20 in effect):

N60 G01 X13.56 W-2.5 F0.013

This example for a CNC lathe shows a tool path motion, where the cutting tool has to reach diameter of 13.56 inches and - *at the same time* - has to move 2.5 inches *into* the negative Z axis direction, represented by incremental designation address W. G90 or G91 is not normally used, when *Group A* of G codes is used (this is the most common G code group). G-code groups are described on *page 55*.

Anytime there is a switch between absolute and incremental mode in a CNC program, part programmer must be careful not to remain in the '*wrong*' mode longer than needed. The switch between modes is usually temporary, for specific purpose and may affect one or more blocks. Always reinstate the original program setting. Remember that both absolute and incremental modes are *modal* - they remain in effect until canceled by the opposite mode.

C. DIAMETER PROGRAMMING

Every X-axis coordinate programmed for CNC lathes can be entered as a *diameter* value. This approach simplifies lathe programming and makes the program easier to read. Normally, the default of virtually all Fanuc controls is set to *diameter programming*. Control system parameter can be changed to interpret the X-axis as a radius input:

G00 X45.0 *Diameter dimension ...when set by parameter*

G00 X22.5 *Radius dimension ...when set by parameter*

Either value is correct, with the appropriate parameter setting (actual parameter number varies for different controls). Diameter programming is easier to understand by both the programmer and operator, because drawings use diameter dimensions for cylindrical parts and measuring diameters at the machine is equally common. Exercise certain caution - if the diameter programming is used, all tool offsets relative to X-axis must be treated as applicable to the *part diameter*, not to its single side (radius value).

Another important programming consideration is the selection of absolute or incremental mode of dimensional input. In diameter programming, where the X-axis amount represents part diameter, it is much more common to program in absolute mode. In those cases, when an incremental value is required, remember that all incremental dimensions in the program must also be specified *per diameter*, not per radius. In incremental mode, the intended X-axis motion will be programmed with the U address, specified as a distance and direction to travel on *diameter*.

For example, both sections of the following metric programs are identical - note that they both *start* in absolute mode and only the diameter inputs appear different:

Example 1 - Absolute diameters:

G00 G42 X85.0 Z2.0 T0404 M08 (ABSOLUTE START)

G01 Z-24.0 F0.3 X95.0

Z-40.0

X112.0

Z-120.0 X116.0 G00 ..

Example 2 - Incremental diameters:

G00 G42 X85.0 Z2.0 T0404 M08 (ABSOLUTE START) G01 Z-24.0 F0.3

U10.0 Z-40.0 (X95.0)

U17.0 (X112.0)

Z-120.0

U4.0 G00 .. (X116.0)

Absolute mode start is necessary - distance is not known.

D. MINIMUM MOTION INCREMENT

Minimum increment (also called the *least* increment) is the *smallest amount of an axis movement* the control system is capable of supporting. The minimum increment is the smallest

amount that can be programmed within the selected dimensional input. Depending on the dimensional input selection, the minimum axis motion increment is expressed either in *millimeters* for the metric system or in *inches* for the imperial system.

Units' system	Minimum increment
Metric	0.001 mm
Imperial	0.0001 inch

In the definition of minimum increment, the most common increments are 0.001 mm and 0.0001 inches for metric and imperial units respectively. For a typical CNC lathe, the minimum increment for the X-axis is also 0.001 mm or 0.0001 inches but is measured on the *diameter* - that means a 0.0005 mm or 0.00005 inches minimum increment per side. Fine tuning for machining precision is much more flexible and precise in metric system than in imperial system of units:

Minimum increment	Converted equivalent
0.001 mm	0.00003947 inches
0.0001 inches	0.00254 millimeters

For high precision work, using metric units should be the preferred dimensional system for part programming. In fact, the metric system is *154% more accurate* than the imperial system, which makes the imperial system almost *60.63% less accurate* than the metric system.

E. FORMAT OF DIMENSIONAL INPUT

The year of 1959 is often considered to be the first year when numerical control technology was first used in practical manufacturing. Since that time, several major changes have taken place that influenced programming format of dimensional input. Even to this day, dimensional data can be programmed in one of four possible ways:

Full address format

Leading zero suppression

Trailing zero suppression

Decimal point

To understand the format differences, looking back some years may be beneficial. Older control systems (mainly the old NC systems as compared to the modern CNC systems) were not able to accept the highest input level of dimensions - *decimal point format* - but all newest controls accept *all* earlier program formats, even when the decimal format is most common. The reason is compatibility with existing programs (old programs). Since decimal point programming method is the latest of four available, control systems that allow decimal point programming can also accept programs written many years earlier (assuming that the control and machine tool are also compatible). *The reverse is not true.*

This is a very important issue, because knowing how the control interprets a *number that has no decimal point* is critical for all tool motion commands and feed rates.

Full Address Format

The full format of a dimensional address is described by the notation of ± 53 in metric units and 44 in imperial units. That means all eight available digits have to be written for the axis words X, Y, Z, I, J, K, etc. For example, metric dimension of 0.42 mm, when applied to the X axis, would

be programmed as:

X00000420 ...*eight-digit format*

Imperial dimension of 0.625, also when applied to the X axis, would be programmed as:

X00006250 ...*eight-digit format*

The full format programming is applicable only to the *very early* control units but is still correct even today. The programmed axis was usually written *without* the axis designation, which is determined by *position* of the dimension within the block. For modern CNC programming, the full format is obsolete and is used here only for reference and comparison. Yes, this format will work quite nicely in modern programs, but don't use it as a standard.

Zero Suppression

Zero suppression concept is a great improvement over the full programming format. It was the adaptation of a new format that reduced the number of zeros in the dimensional input. Many modern controls still support the method of zero suppression, but only for reasons of compatibility with old and proven programs.

Zero suppression means that *either* the *leading* or the *trailing* zeros within the eight-digit dimensional input do not have to be written in the program. The result is a great reduction in program length. Any default setting is done by the control manufacturer, although it can be optionally set by a system parameter. *Don't make any changes without a valid reason!*

Since *leading* zeros suppression and *trailing* zeros suppression are mutually exclusive, which one is more practical for addresses without a decimal point? As it depends on parameter setting of the control system *or* status designation by the control manufacturer, the actual control status must always be known. This status determines which zeros can be suppressed. It may be zeroes at the *beginning* or zeros at the *end* of a dimension without a decimal point. These zeros are *before the first significant digit* or *after the last significant digit*. In the extremely unlikely event that a CNC system is equipped with zero suppression feature as the only mode, programming with decimal point will not be possible. To illustrate the results of zero suppression, earlier examples will be used.

The metric units input of 0.42 mm, also applied to the X axis, is programmed with leading zeros suppressed, as:

X420

The same dimension of 0.42 mm with trailing zeros suppressed, will appear in the program as:

X0000042

If the imperial input of 0.625 inches is to be programmed with *leading zero suppression* format in effect and applied to the X axis, it will appear in the program as:

X6250

The same dimension of 0.625 inches with *trailing zeros suppressed*, will appear in the program as:

X0000625

Although both examples above illustrate only one small application, the impression that leading zero suppression is more practical than trailing zero suppression is accurate. Many older control systems are indeed set arbitrarily to accept *leading zero suppression as the default*, because of its practicality. Here is the reason why - study it carefully, although today the subject is more trivial than practical. On the other hand, if even one decimal point is omitted (forgotten, perhaps) in the part program, this knowledge becomes very useful, and the subject is not trivial anymore.

Preference for Leading Zero Suppression

Minimum and maximum dimensional input the control system can accept consists of eight digits, without a decimal point, ranging from 00000001 to 99999999:

Minimum: 0000.0001 inches or 00000.001 mm

Maximum: 9999.9999 inches or 99999.999 mm

The decimal point is *not* written. If the program uses zero suppression of either type, a comparison of input values should be useful:

Input value comparison - Metric units (millimeters)		
Decimal point	Leading zeros suppression	Trailing zeros suppression
X0.001	X1	X00000001
X0.01	X10	X0000001
X0.1	X100	X000001
X1.0	X1000	X00001
X10.0	X10000	X0001
X100.0	X100000	X001
X1000.0	X1000000	X01
X10000.0	X10000000	X1

Leading zero suppression is more beneficial, because it supports numbers with the more practical *small* fractional part (fine sizes) rather than those with a large integer part.

For imperial input, the results will be similar:

Input value comparison - Imperial units (inches)		
Decimal point	Leading zeros suppression	Trailing zeros suppression
X0.0001	X1	X00000001
X0.001	X10	X0000001
X0.01	X100	X000001
X0.1	X1000	X00001
X1.0	X10000	X0001
X10.0	X100000	X001
X100.0	X1000000	X01
X1000.0	X10000000	X1

Even if programs always use decimal point, knowing the effect of zero suppression is important. For example, what will happen if the programmer forgets a decimal point in a motion address or the CNC operator forgets to punch it in for an offset entry? These are serious - *and common* - errors that can be avoided with some care and good knowledge.

To complete the section on zero suppression, let's look at a program input that uses an axis letter but *not as a coordinate word*. A dwell command is suitable for explanation. Starting on *page 175*, all details relating to dwell programming are covered. For now, just use the basic format and one second dwell time as the unit. The dwell format is specified by the dwelling axis X, as being X5.3. This format tells us that the dwell can be programmed with the X axis, followed by the maximum of eight digits, always positive. If the control system allows decimal point input, there is no confusion. If leading or trailing zeros have to be suppressed, the programmed input is very important.

For example, a program requires dwell lasting 0.5 of a second (one half of a second). In various formats, the program block containing the ½ sec. dwell will be:

Full format	X00000500	
No leading zeros	X500	
No trailing zeros	X000005	
Decimal point	X0.5 or	X.5

Note that logic behind the format is the same for dwell as for the coordinate words. The programmed format will always adhere to the address notation. Incidentally, in several fixed cycles, the dwell is expressed by the P address, which *does not* take a decimal point at all and always must be programmed with leading zero suppression mode in effect. Half a second dwell will be equal to P500. More on using dwell in fixed cycles, *see page 183*.

Decimal Point Programming

All modern programming will use the decimal point for dimensional and some other input. Programming decimal point, particularly for program data requiring very accurate measurement, makes the CNC program much easier to develop and to interpret at a later date.

From all available program addresses that can be used, not all can be programmed with the decimal point. Those that can are those that specify data in millimeters, inches, or seconds (some exceptions do exist).

The following two lists contain addresses where the decimal point is allowed in programs for both milling and turning controls:

Milling control programs: X, Y, Z, I, J, K, A, B, C, Q, R Turning control programs:

X, Z, U, W, I, K, R, C, E, F (also B and Y)

Control system that supports the decimal point method of programming, can also accept dimensional values *without* a decimal point, in order to allow compatibility with older programs. In such cases, it is important to understand the principles of programming format using leading and trailing zeros. If they are used correctly (see earlier explanations), there will be no problem to apply various dimensional formats to any other control system, old or new. If possible, always use decimal point as a standard approach.

This compatibility enables many long-time users to load their old programs into newer CNC controls - but *not* the other way around - usually with only minor modifications, or no modifications at all.

Virtually all modern CNC units do not have the ability to accept a paper tape because they have no tape reader. To convert any tapes that contain good programs, there are two options - one, have someone to install a tape reader in the control, if possible and justified (probably not). The other method is to store the contents of a tape in a personal computer. This method is very inexpensive and offers much better storage options than a paper tape. With suitable software and a portable tape reader (that is the catch), the task is not impossible. Keep also in mind that there are few companies specializing in this kind of work.

Dimensional data in the metric system assume 0.001 mm minimum increment, while in the imperial system the increment is 0.0001 of an inch (leading zero suppression mode is in effect as a default) - therefore:

Y12.56 is Y125600 ...for imperial units Y12.56 is Y12560 ...for metric units

Programmed values with and without a decimal point can be mixed together in the same block:

N230 X4.0 Y-10

This may be beneficial for extreme conservation of system memory. For example, the X4.0 word will require fewer characters than the word X40000 - on the other hand, the Y-10 is shorter than the decimal point equivalent of Y-0.001 (both examples are in imperial units). If all digits *before* or *after* the decimal point are zeros, they do not have to be written:

X0.5 = X.5

Y40.0 = X40.

Z-0.1 = Z-.1 F12.0 = F12.

R0.125= R.125 ...etc.

Any zero value *must* be written - for example, X0 cannot be written as X only. In this digital book, all program examples will use the decimal point format, whenever possible. Many programmers prefer to program zeros as in the left column of the example. This method does add a few characters into the system memory, but program data are much easier to read. They are also easier for learning.

Input Comparison

Differences in the input format for both metric and imperial dimensioning can be seen clearly. One more time, the same examples will be shown, as before:

Metric example - input of 0.42 mm:

Full format X00000420

No leading zeros X420

No trailing zeros X0000042

Decimal point X0.42 or X.42

Imperial example - input of 0.625 inches:

Full format X00006250

No leading zeros X6250

No trailing zeros X0000625 Decimal point X0.625 or X.625

F. CALCULATOR TYPE INPUT

In some specialized industries, such as woodworking or fabricating, the majority of dimensions (especially metric) do not require decimal parts, only whole numbers. In these cases, the decimal point would always be followed with a zero. Fanuc provides a solution to such situations by a feature called the *calculator input*. Using this feature can shorten overall program size, often significantly.

Calculator type input requires setting of a system parameter. Once the parameter is set, neither the decimal point nor the trailing zeros have to be written - they will be assumed. For example, X25 will be interpreted as X25.0, *not* the normally expected X0.025 (mm) or X0.0025 (inches).

In case the input value does require decimal point, it can be written as usually. This means all values with a decimal point will be interpreted correctly and numbers without the decimal point

will be treated as major units only (millimeters or inches). Here are some examples:

Standard Input	Calculator Input
X345.0	X345
X1.0	X1
Y0.67	Y0.67
Z7.48	Z7.48

Normally, the control system is set to leading zero suppression mode and non-decimal values are interpreted as the number of smallest units. For example, Z1000 in G21 mode will be equivalent to Z1.0 (mm), in G20 mode as

Z0.1 (inches).

11. SPINDLE CONTROL

Both types of CNC machines, machining centers and lathes, use spindle rotation when removing excessive material from a part. Spindle rotation may be that of the cutting tool (milling) or the part itself (lathes). In both cases, activities of machine spindle and working feed rate of the cutting tool need to be strictly controlled by the program. These CNC machines require instructions that relate to the selection of a suitable *speed* of machine spindle and a cutting *feed rate* for a given job.

There are several ways to control machine spindle and cutting feed rate and they all depend mainly on the type of CNC machine and the current machining application. In this chapter, we look at the *spindle control* and its programming applications.

A. SPINDLE FUNCTION

Program command relative to spindle speed is controlled in the CNC system by the address S. Programming format of the S address is usually within the range of 1 to 9999 for standard machines and no decimal point is allowed:

S1 to S9999

For many high-speed CNC machines is not unusual to have spindle speed available up to five digits, in the range of 1 to 99999, within the S address range:

S1 to S99999

Maximum spindle speed range available in the control must always be *greater* than the maximum spindle speed range of the machine itself. It is quite typical that virtually all control systems support a much greater range of spindle speeds than the CNC machine allows. In programming spindle speeds, the limitation is always caused by the machine unit, *not* by the control system.

Spindle Speed Input

The address S relates to machine spindle function and must always be assigned a specific numeric value in CNC programs. There are three alternatives as to what numeric value (input) of the spindle function may be:

Code number	old controls only - now obsolete
Spindle speed	r/min
Cutting speed	m/min or ft/min

On CNC lathes, both modern alternatives may exist, depending on the control system. For CNC milling systems, cutting speed is not applicable (except for calculations), but direct spindle speed is. Spindle speed selection by special code number is an obsolete concept, not required on modern controls and not discussed here.

Spindle speed designation S is not sufficient to be programmed by itself. In addition to the selected spindle speed address, certain additional attributes are necessary as well. These are attributes that control spindle function environment. For example, if the spindle speed is specified as S400 in the program, programming instruction is not complete, because the spindle function stands by itself. It does not include *all* information the control system requires for spindle data. A spindle speed value that is set, for example, to 400 r/min or 400 m/min or 400 in/min or 400 ft/min (depending on actual machining application), still does not contain all necessary information, namely, the spindle rotation *direction*.

Most machine spindles can be rotated in two directions *clockwise* or *counterclockwise*, depending on the type and setup of the cutting tool used. Spindle rotation has to be specified in the program, *in addition* to the spindle speed function. There are two miscellaneous functions provided

by the control system that control direction of the spindle -

B. M03 AND M04. DIRECTION OF SPINDLE ROTATION

Thinking in terms of *right* and *left*, *up* and *down*, *clockwise* and *counterclockwise*, and similar directional terms, is thinking in terms that are *relative* to some known reference. To describe a spindle rotation as clockwise (CW), or as counterclockwise (CCW), some established and standard reference method is needed, in this case a reference point of view (reference viewpoint).

Direction of spindle rotation is always relative to the point of view that is established from the spindle side of the machine. This part of a machine that contains the spindle and is generally called the machine headstock. Looking from the machine headstock area into *the direction along spindle center line and towards its face*, establishes the correct viewpoint for defining CW and CCW rotation of the spindle. For CNC drills, milling machines and CNC machining centers, the reference point of view is quite simple to understand. For CNC lathes, the rules are exactly the same, and will be described shortly.

Direction for Turning

A comparable approach would also seem logical for CNC lathes. After all, the operator also faces the front of a machine, same as when facing a vertical machining center. *Figure 12-2* shows a front view of a typical CNC lathe.

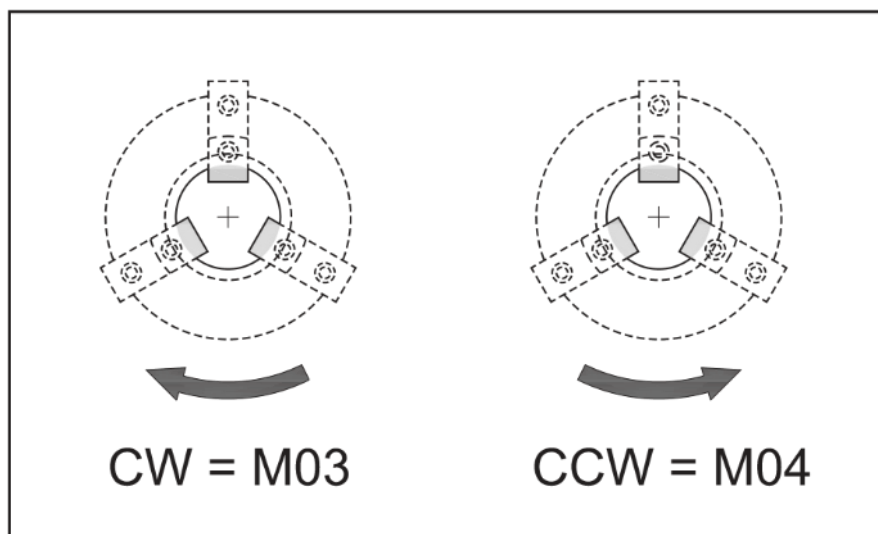
Although descriptions CW and CCW in the illustration appear to be opposite to the direction of arrows, they are correct. The reason is that there are two possible points of view, and they are both using spindle center line as the viewing axis. Only one of the viewpoints matches the standard definition and is, therefore, correct. The definition of spindle rotation for lathes is *exactly* the same as for machining centers.

To establish spindle rotation as CW and CCW, look from the headstock area towards the spindle face

The first and proper method will establish the relative viewpoint starting *at the headstock* area of the lathe. From this position, looking towards the tailstock area, or into the same general area, clockwise and counterclockwise directions are established correctly.

The second method of viewing establishes the relative viewpoint starting *at the tailstock* area, facing the chuck. This is an *incorrect* view to establish spindle rotation!

Compare the following two illustrations - *Figure 12-3* shows the view from the headstock, *Figure 12-4* shows the view from the tailstock and arrows must be reversed.



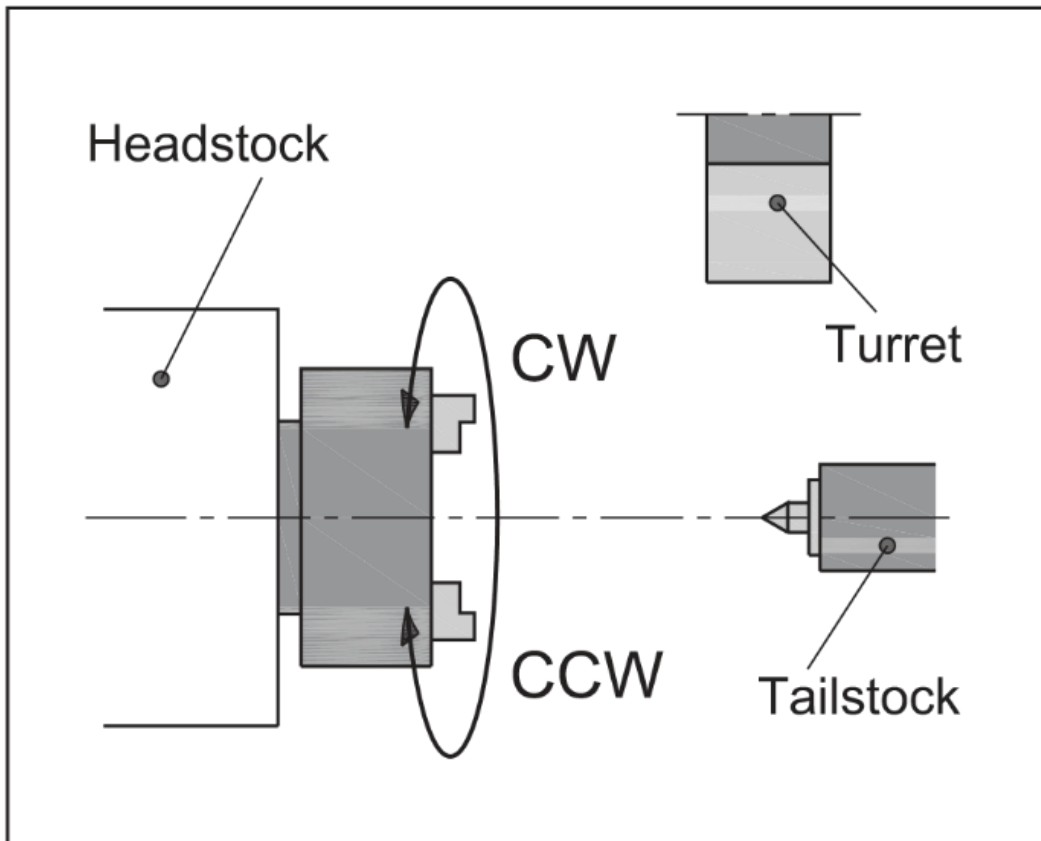


Figure 12-2

Typical view of a slant bed two axis CNC lathe.

CW and CCW directions only appear to be reversed Figure 12-3

Spindle rotation direction as viewed from the headstock

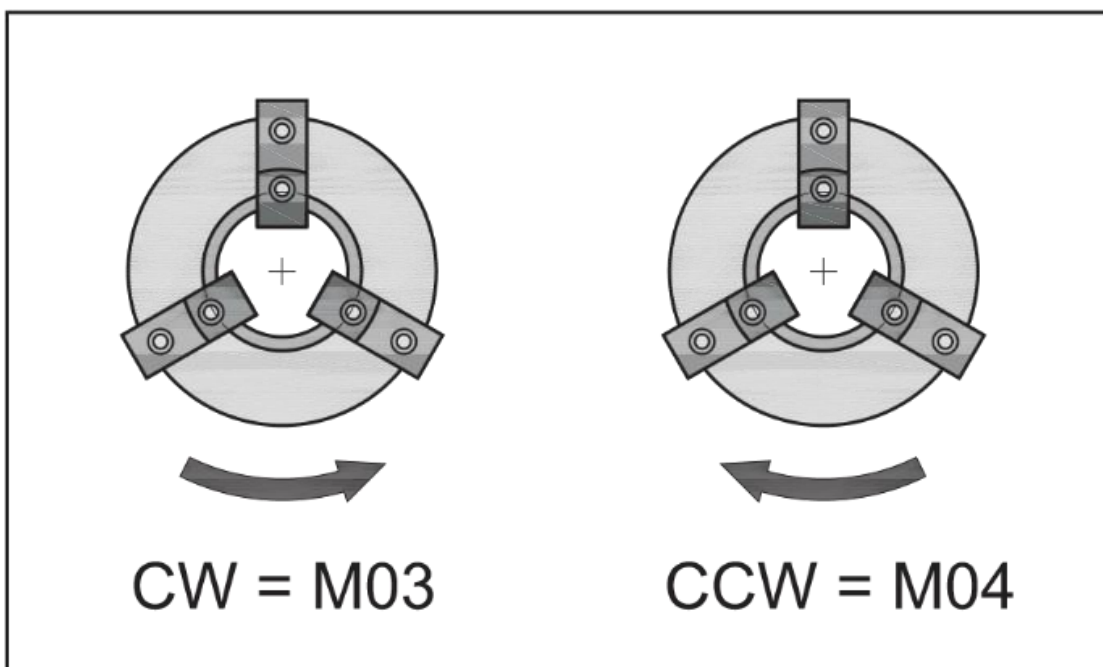


Figure 12-4

Spindle rotation direction as viewed from the tailstock

Direction Specification

If the spindle rotation is clockwise, M03 function is used in the program - if spindle rotation is

counterclockwise, M04 function is used in the program.

Since spindle speed S in the program is dependent on the spindle rotation function M03 or M04, their relationship in a CNC program is very important.

Spindle speed address S and spindle rotation function M03 or M04 must always be accepted by the control system together. One without the other will not mean anything to the control.

There are at least two correct ways to program spindle speed and spindle rotation:

If spindle speed and rotation are programmed together in the same block, spindle speed and spindle rotation will start simultaneously

If spindle speed and rotation are programmed in separate blocks, spindle will not start rotating until both the speed and the rotation commands have been processed

Spindle Startup

The following examples demonstrate a number of correct starts for spindle speed and spindle rotation in a program. All examples assume that there is *no active setting* of spindle speed S, either through a previous program or through *Manual Data Input* (MDI). On CNC machines, there is no registered or default spindle speed when machine power is turned on.

Example D - Turning application with G50:

N1 G20

N2 G50 X13.625 Z4.0 T0100 N3 G96 S420 M03 (SPEED SET - ROTATION STARTS) N4

...

This is the preferred example for CNC lathes, if the older G50 setting method is used. Because the spindle speed is set as cutting speed CSS - *Constant Surface Speed*, the control system will calculate the actual revolutions per minute (r/min) based on the CSS setting of 420 (ft/min) and the current part diameter at X13.625. The next example E is correct but not recommended (see caution box above).

Example E - Turning application with G50:

N1 G20

N2 G50 X13.625 Z4.0 T0100 M03 (ROTATION SET)

N3 G00 X6.0 Z0.1 (NO ROTATION) N4 G96 G01 Z0 F0.04 T0101 S420 (ROTAT. STARTS) N5

Example F - Turning application without G50:

N1 G20 T0100

N2 G96 S420 M03 (SPEED SET - ROTATION STARTS) N3 G00 ...

In this more contemporary example (G50 is not used as a position register command anymore), machine spindle speed will be calculated for a tool offset value stored in the *Geometry Offset* register of the control system. Control system will perform the calculation of actual r/min when the block N2 is executed.

These examples are only *technically correct* methods for a spindle start. All contain selected rotation at the beginning of a program and cover both, milling and turning applications. The example for *beginning* of a program has been selected intentionally, because for any first tool in the program, there is no active speed or rotation in effect (normally carried on from a previous tool). However, the control unit may still store spindle speed and rotation from the last tool of the *previous* job!

Any tool *following* the first tool will normally assume the programmed speed selection and rotation for the previous tool. If only the spindle speed command S is programmed for the next

tool, *without* rotation direction, the tool will assume the *last* programmed rotation direction. If only the rotation direction code M03 or M04 is programmed, spindle speed S will be the same as for the previous tool.

Be careful if a program contains program stop functions M00 or M01, or the spindle stop function M05. Any one of them will automatically stop the spindle. It means to be absolutely sure as to *when* the spindle rotation will take place and what it will be. Always program spindle speed selection and its rotation in the same block and for *each* tool. Both functions are logically connected and placing them within a single block will result in a coherent and logical program structure.

C. SPINDLE STOP

Normally, most work requires a spindle that rotates at a certain speed. In some cases, a rotating spindle is not always desirable. For example, before programming a tool change or reverse a part in the middle of a program, the spindle must be stopped first. Spindle must also be stopped during a tapping operation and at the end of program. Some miscellaneous functions will stop spindle rotation automatically (for example, functions M00, M01, M02 and M30). Spindle rotation will also stop automatically during certain fixed cycles. For a total control of a program, spindle stop should always be specified in the program. Counting on other functions to stop the spindle is not a good programming practice. There is a special function available in programming, to stop the spindle.

In order to stop the spindle rotation, use function M05. M05 will stop both clockwise and counterclockwise spindle rotation. Because M05 does not do anything else (unlike other functions that also stop the spindle, such as M00, M01, M02, M30 and others), it is used for situations, where the spindle must be stopped *without* affecting any other programmed activities. Some typical examples include reversal in tapping, tool motion to the indexing position, turret change position, or after machine zero return, depending on application. Using one of the other miscellaneous functions that automatically stop spindle, the M05 function is not required. On the other hand, it does no harm to program exactly what is required, in a particular order.

This method may result in a slightly longer program, but it will be easier to read and maintain it, mainly by CNC operators with limited experience.

Spindle stops function can be programmed as a separate block as well, for example:

N120 M05

or in a program block containing a tool motion, such as in
the next example:

N120 Z1.0 M05

Tool motion will always be completed first, *then* the spindle will be stopped. This is a safety feature built into the control system. Always remember to program M03 or M04 to reinstate spindle rotation.

D. SPINDLE ORIENTATION

The last M-function that also relates to spindle activity, is M19. This function is most commonly used to set a machine spindle into an oriented position. Other M-codes may be valid, depending on the control system, for example M20 on some controls. Spindle orientation function is a very specialized function, seldom appearing in the program itself. When M19 function is used, it is mainly during setup, in *Manual Data Input* mode (MDI). This function is

generally common for milling systems, because only specially equipped CNC lathes will require it. M19 function can only be used when the spindle is stationary, usually after the spindle has stopped. When the control system executes M19 function, the following action will result:

The spindle will make a slight turn in both directions, clockwise and counterclockwise, and after a short period, the internal locking mechanism will be activated. In some cases, the locking that takes place is audible. The spindle will be locked in a *precise* position, and rotating it by hand, will not be successful. The exact locking position is fixed and is determined by the machine tool manufacturer, indicated by the setting angle - *Figure 12-5*.

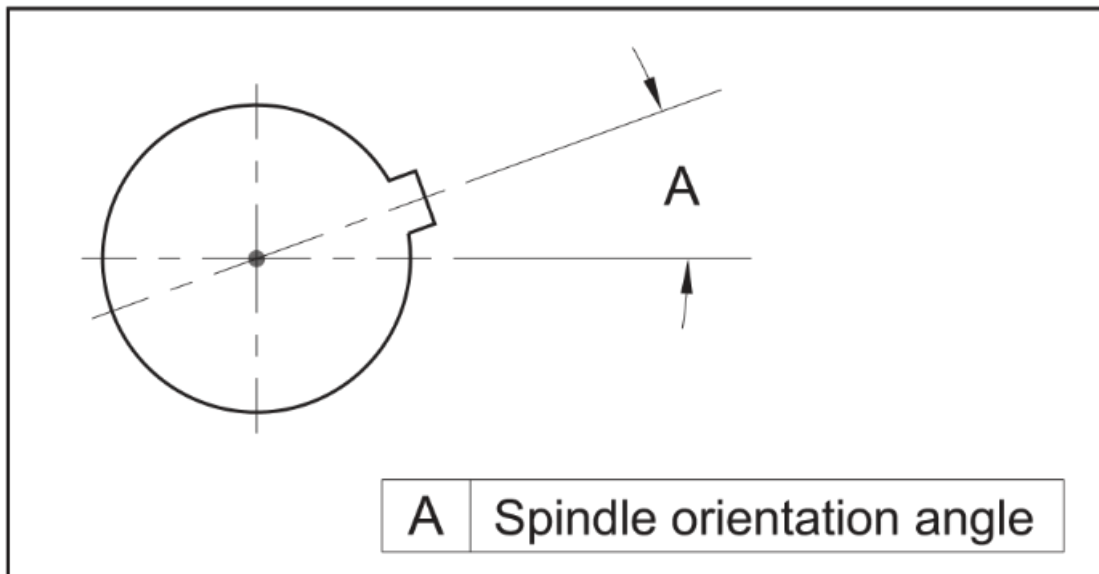


Figure 12-5

Spindle orientation angle is defined by the machine manufacturer and cannot be changed

In normal CNC operation, M19 function enables the machine operator to place a tool into the spindle manually and guarantees a proper tool holder orientation. Later chapters will provide more details about spindle orientation and its applications, for example, in single point boring cycles.

WARNING - An incorrect tool holder orientation may result in a damage to the part or machine

Many CNC machining centers (not all) use tool holders that can be placed into the tool magazine only one way. To achieve this goal, the tool holder has a special notch built-in, that matches the internal design of spindle - *Figure 12-6*. In order to find side of the holder that has the notch, usually there is a small dimple or other mark on the notch side. This design is intentional.

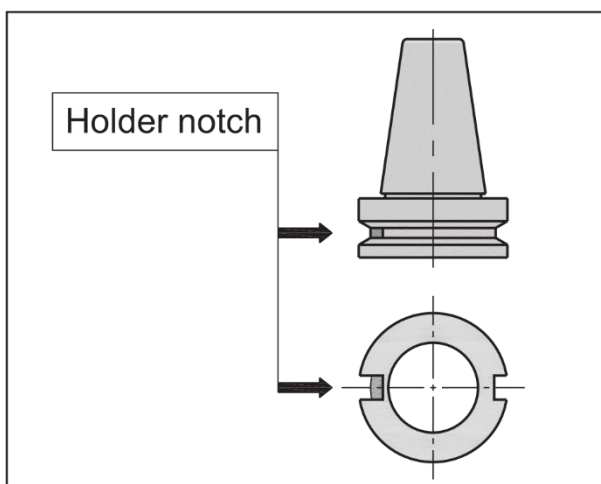


Figure 12-6

Built-in notch in a tool holder used for correct tool orientation in the spindle - not all machines require this feature

For tools with several flutes (cutting edges), such as drills, end mills, reamers, face mills, etc., the orientation of cutting edge relative to the stopped spindle position is not that important.

However, for single point tools, particularly boring bars, orientation of the cutting-edge during setup is extremely important, particularly when certain fixed cycles are used. There are two fixed cycles that use the *built-in* spindle orientation - G76 and G87 - in both, the tool retracts from the machined hole *without rotating*. In order to prevent damage to the just finished bore, the tool retraction must be controlled. Spindle orientation guarantees that the tool will shift away from the finished bore into a clear direction. An accurate initial setup is necessary!

Those machines that allow placing tool holder into the spindle either way still require proper setting of tools that shift when G76 or G87 fixed cycles are programmed.

E. SPINDLE SPEED - R/MIN

When programming CNC machining centers, designate the spindle speed directly in *revolutions per minute* (r/min). A basic block that contains spindle speed of 200 r/min, for example, will require this data entry:

N230 S200 M03

Such format is typical to milling controls, where no cutting speed (peripheral or surface speed) is used. There is no need to use a special preparatory command to indicate the r/min setting, it is the control default. The r/min setting must have a minimum increment of one. Fractional or decimal values are not allowed, and the r/min must always be set within the range of any machine specifications.

A few machining centers may be equipped with the option of a *dual* spindle speed selection - *direct r/min* and *cutting speed*. In this case, as well as for all lathe programming, a proper preparatory command is used to distinguish *which* selection is active. G96 is used for cutting speeds, G97 for direct specification of r/min. The distinction between them is discussed next.

F. SPINDLE SPEED - SURFACE

Programmed spindle speed should be based on the machined *material* and the *cutting tool diameter* (machining centers), or the *part diameter* (lathes). General rule is that the larger the diameter, the slower the spindle r/min must be. Spindle speed should never be guessed - *it should always be calculated*. Such a calculation will guarantee that spindle speed is *directly proportional to the programmed diameter*. An incorrect spindle speed will have a negative effect on both tool and part.

Material Machinability

To calculate spindle speed, each work material has a suggested machinability rating for a given tool material. This rating is either a percentage of some common material, such as mild steel, or a direct rating in terms of its *cutting (peripheral) speed* also known as *surface speed*. Surface speed is specified in *feet per minute (ft/min)* in imperial units, and in *meters per minute (m/min)* in metric system. An older abbreviation used for *ft/min* is *FPM*, meaning *Feet Per Minute*. Amounts of surface speeds indicate the level of machining difficulty with a *given tool material*. The lower the surface speed, the more difficult it is to machine the work material.

Note the words '*given tool material*'. To make all comparisons meaningful and fair, they must be made with the same type of cutting tool, for example, surface speeds for high-speed steel tools will be much lower than those for cobalt-based tools and, of course, for carbide tools.

Based on the surface speed and the cutter diameter (or part diameter for lathes), machine spindle speed can be calculated in *revolutions per minute*, using one mathematical formula for imperial units system and another when metric units are programmed.

Surface speed will increase for soft materials and decrease for hard materials

High speed steel tools will run slower than carbide tools in the same material

Spindle Speed - Imperial Units

To calculate spindle speed in *r/min*, the *surface* speed of the material for the cutting tool type must be known, as well as the part or tool diameter:

	12 ft / min
r / min	_____
	D

+ where . . .

r/min = Spindle speed in revolutions
per minute

12 Multiplying factor - feet to
= inches conversion

ft/min Surface speed in feet per
minute

= Constant 3.1415927

D Diameter in inches (cutter
= meter for milling, or part
meter for turning)

Example:

Surface speed for selected material is 150 ft/min, and the cutting tool diameter is 1.75 inches:

$$r/min = (12 \ 150) / (3.1415 \ 1.75)$$

$$= 327.4$$

$$= 327 \text{ r/min}$$

Many programming applications can use a shorter formula, without losing any significant accuracy:

	3 82. ft / min
r / min	_____
	D

For less demanding calculations, the 3.82 constant may be rounded to 4 and used as an easier calculation without a calculator. The measuring units must be applied properly, otherwise the results will not be correct.

Never mix imperial and metric units in the same program!

Spindle Speed - Metric Units

When metric system is used in a program, the logic of previous formula is the same, but the units are different:

	1000 m / min
r / min	_____
	D

+ where . . .

$$r/min = \text{Spindle speed in revolutions per minute} \times 1000 = \text{Multiplying factor} \times$$

meters to mm conversion m/min = Surface speed in m/min

$$= \text{Constant } 3.1415927$$

D = Diameter in *mm* (cutter diameter for milling,

or part diameter for turning)

Example:

Given surface speed is 30 m/min and the cutting tool diameter is 15 mm:

$$r/\text{min} = (1000 \text{ } 30) / (3.1415 \text{ } 15)$$

$$= 636.6$$

$$= 637 \text{ r/min}$$

A shorter version of the formula is an acceptable alternative and almost as accurate as the precise formula:

ω / min 3183.	D

Again, by replacing the constant 318.3 with constant 320 (or even 300), the r/min will be somewhat inaccurate, but most likely within an acceptable range.

G. CONSTANT SURFACE SPEED

On CNC lathes, machining process is different from milling process. Turning tool has no diameter and the diameter of a boring bar has no relationship to the spindle speed. It is the *part diameter* that is the diameter used for spindle speed calculations. As the part is being machined, its diameter changes constantly. For example, during a facing cut or during roughing operations the diameter changes - see illustration in *Figure 12-7*. Programming the spindle speed in r/min is not practical - after all, which of the numerous diameters should be selected to calculate the r/min? The solution is to use the *surface speed* directly, as a block entry in the lathe program.

To select surface speed is only a half of the procedure. The other half is to communicate this selection to the control system. Control has to be set to the *surface speed* mode, not the *spindle speed (r/min)* mode. Operations as drilling, reaming, tapping, etc., are common on a lathe and they require the direct *r/min* in the program. To distinguish between the two alternatives in lathe programming, the choice of *surface speed* or *revolutions per minute* must be specified. This is done with preparatory commands G96 and G97, *prior to* the spindle function:

G96 S. . M03 Surface speed selected (m/min or ft/min) **G97 S. . M03** Spindle speed(r/min) selected

For milling, this distinction normally does not exist and spindle speed in r/min is always assumed.

By programming the surface speed command G96 for turning and boring, the control enters into a special mode, known as the *Constant Surface Speed* or CSS. In this mode, actual spindle revolutions will *increase* and *decrease* automatically, depending on the actual diameter being cut (current diameter). Automatic *Constant Surface Speed* feature is built in the control systems available for all CNC lathes. It is a feature that not only saves programming time, but it also allows the tool to remove *constant amount of material* at all times, thus saving the cutting tool from excessive wear and creating a better surface finish.

Figure 12-7 shows a typical example, when a facing cut starts at X6.2 (6.2) and faces the part to machine centerline or slightly below. G96 S375 was used, and 6000 r/min was the maximum

spindle speed of the lathe.

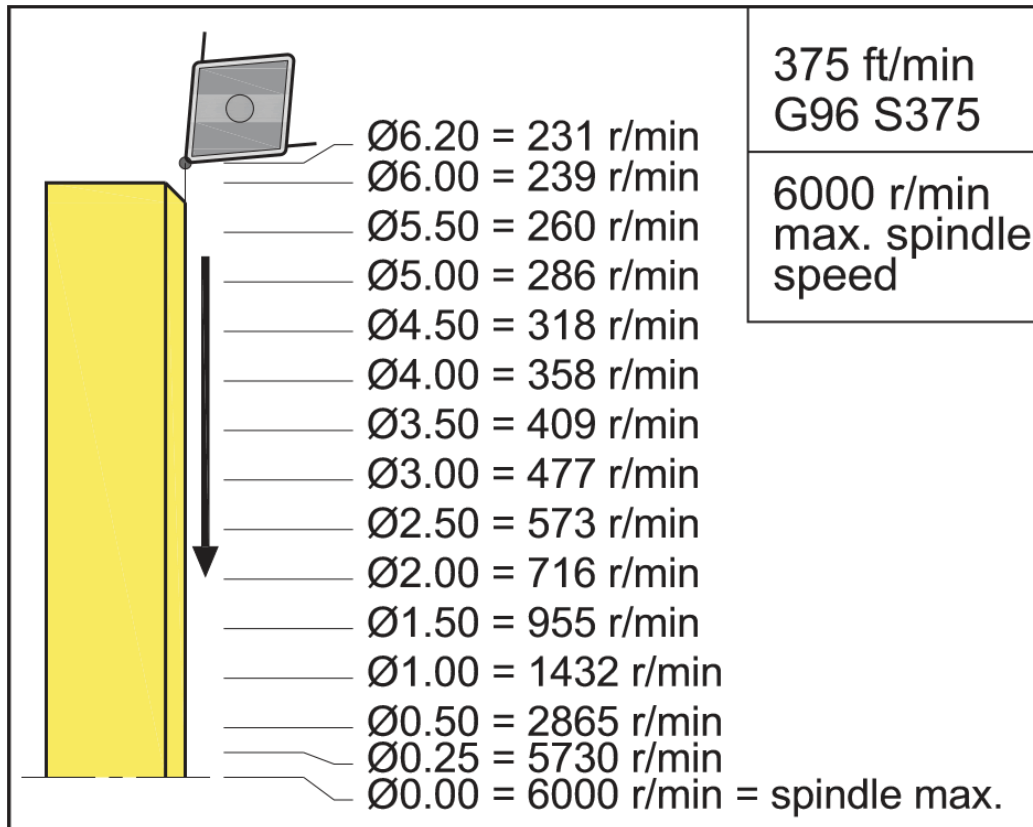


Figure 12-7

Example of a facing cut using constant surface speed mode G96

Although only selected diameters are shown in the illustration, along with their corresponding revolutions per minute, the updating process is always constant. Note the sharp increase in r/min as the tool moves closer to machine centerline. When the tool reaches X0 (0.0), the speed will be at its *maximum*, within the current gear range. As this speed may be extremely high in some cases, control system allows setting of a certain maximum, described later.

To program a surface speed for a CNC lathe, there are several options. In the following three examples, the most important ones will be examined. Gear change functions (if available) are omitted for all examples.

Example 1:

Surface speed is set right after the coordinate setting is read from *Geometry* offset:

N1 G20

N2 (GEOMETRY OFFSET SET TO X16.0 Z5.0) T0100

N3 G96 S400 M03 ...

In this common application, the actual spindle speed will be based on the preset diameter of 16 inches, resulting in 95 r/min in block N3. In some cases, this will be too low. Consider another example:

Example 2:

On very large CNC lathes, geometry offset setting of the X-axis diameter is quite large, say 24.0 inches. In the previous example, target diameter of the next tool motion was not important, but in this case it is. For example:

N1 G20

N2 (GEOMETRY OFFSET SET TO X24.0 Z5.0) T0100

N3 G96 S400 M03 N4 G00 X20.0 T0101 M08 ...

In *Example 2*, the initial tool position is at X24.0 and the tool motion terminate at X20.0, both values are diameters. This translates to an actual motion of only 2.0 inches. At the X24.0, the spindle will rotate at 64 r/min, at X20.0 it will rotate at 76 r/min. The difference is very small to warrant any special programming. It is different, however, if the starting position is at a large diameter, but a tool moves to a much smaller target diameter.

Example 3a:

From the initial position of 24.0 inches, the tool will move to a rather small diameter of 2.0 inches:

N1 G20

N2 (GEOMETRY OFFSET SET TO X24.0 Z5.0) T0100

N3 G96 S400 M03 N4 G00 X2.0 T0101 M08 ...

Spindle speed at the start of program (block N3) will be the same as in previous example, at 64 r/min. In the next block (N4), the speed calculated for 2.0 inch will be 764 r/min, automatically calculated by the control. This rather large change in spindle speeds may have an adverse effect on some CNC lathe work. What may happen is that the cutting tool will reach the 2.0 inch *before* the spindle speed fully accelerates to the required 764 r/min. The tool may start removing material at a speed much slower than intended. In order to correct the problem, data in CNC program need to be modified:

Example 3b:

Program modification takes place in block N3. Instead of programming constant surface speed mode, program direct *r/min* for the target of 2.0 inches, based on 400 ft/min surface speed. Actual *r/min* has to be calculated first, then the CSS setting will be programmed in a subsequent block:

N1 G20

N2 (GEOMETRY OFFSET SET TO X24.0 Z5.0) T0100

N3 G97 S764 M03

N4 G00 X2.0 T0101 M08

N5 G96 S400 ...

In the example, at 24.0 (X24.0 offset), the actual speed would be only 64 r/min. At 2.0 (X2.0 in N4), the speed will be 764 r/min. Cutting tool *may* reach X2.0 position before the spindle speed has accelerated to full 764 r/min, *if it is not calculated and programmed earlier* - see block N3.

This technique is only useful if the CNC lathe does not support automatic time delay. Many modern lathes have a built-in timer, that forces the cutting tool to wait before actual cutting, until the spindle speed has fully accelerated.

Older CNC lathes used G50 position register command, and the initial position was part of the program. For example, instead of geometry offset set to X24.0 Z5.0, program would contain G50 X24.0 Z5.0. *Geometry Offset* setting is much more flexible, as it is done at the machine.

Maximum Spindle Speed Setting

When CNC lathe operates in the *Constant Surface Speed* mode, the spindle speed is directly related to the current part diameter. The smaller the work diameter is, the greater the spindle speed will be. So, a natural question is - what will happen if the tool diameter is *zero*? It may seem impossible to ever program a zero diameter, but there are at least two cases when that is exactly the case.

In the first case, zero diameter is programmed for all *centerline operations*. All drilling, center drilling, tapping and similar operations are programmed at the zero diameter (X0). These operations are always programmed in the direct r/min mode, using G97 command. In G97 mode, the spindle speed is controlled directly, r/min does not change.

The second case of a zero diameter is when *facing off* a solid part all the way to the centerline. This is a different situation. For all operations at X0, the cutting diameter does not change, because a direct r/min is programmed. During a face cutting operation, the diameter changes all the time while material removal continues until the tool reaches the spindle centerline. No, don't reach for the formulas explained earlier. *Any calculation with a diameter in the formula being zero, will result in error!* Rest assured, there will *not* be 0 r/min at the spindle centerline - or an error - in G96 mode. Return to *Figure 12-7* for illustration.

Whenever the surface speed spindle mode is active and tool reaches spindle centerline at X0, the result will normally be the *highest* spindle rotation possible, *within the active gear range*. It is paradoxical, but that is exactly what will happen. Such situation is acceptable when the part is well clamped, does not extend too far from the chuck or fixture, tool is strong and robust, and so on. When a part is mounted in a special fixture, or an eccentric setup is used, when part has a long overhang, or when some other adverse conditions are present, the maximum spindle speed at the centerline may be *too high* for operating safety!

There is a simple solution to this problem, using a programming feature available for Fanuc and other controls. Surface speed mode can be used with a *preset highest limit*, specified in *revolutions per minute*. Program function for maximum spindle r/min setting is normally G50 or G92 on some lathes. This maximum setting is sometimes called *maximum spindle speed clamping*. Do not confuse this G50/G92 with its other meaning, position register preset. Here is an example of G50 as a speed limiting command:

O1201 (SPINDLE SPEED CLAMP)

N1 G20 T0100

N2 G50 S1500 (1500 R/MIN MAX)

N3 M42 (HIGH SPINDLE RANGE)

N4 G96 S400 M03 (CSS AND 400 FT/MIN)

N5 G00 G41 X5.5 Z0 T0101 M08

N6 G01 X-0.07 F0.012 (BELOW CENTER LINE)

N7 G00 Z0.1

N8 G40 X9.0 Z5.0 T0100 N9 M01

What actually happens in program O1201? Block N1 selects imperial units of measurement and T01. The critical block N2 has *a simple meaning*:

G50 S1500

... means ...

Do not exceed 1500 r/min in G96 surface speed mode

Block N3 selects the spindle gear range; block N4 sets the surface speed mode, using 400 ft/min surface speed. Spindle rotation M03 is called in the same block. In block

N5, the tool makes a rapid motion towards 5.5 and the part front face. During rapid motion, tool nose radius offset, and coolant function are activated. Spindle speed at 5.5 will be 278 r/min, using formula described earlier in this chapter. Next block N6 is the actual facing cut. At the cutting feed rate of 0.012 in/rev, the tool tip faces off the blank part to the centerline. In reality, the cut end point is programmed on the other side of spindle center line (X-0.07).

Tool nose point radius size must be taken into consideration when programming with the tool nose radius offset (G41/G42) and to the machine centerline. A special section later explains what exactly will happen during this cut - see *page 278* for further details.

Block N7 moves the tool tip 0.100 inches away from the face, at a rapid rate. In the remaining two blocks, the tool will be rapid to the indexing position with a cancellation of radius offset in N8 and an optional program stop is provided in block N9.

Now, think of what happens in critical blocks N5 and N6. Spindle will rotate at the speed of 278 r/min at the 5.5. Since the CSS mode is in effect, as the tool tip faces off the part, its diameter is becoming smaller and smaller while the *r/min is constantly increasing*.

Without the maximum spindle speed limit in block N2, the spindle speed at the centerline will be equivalent to the *maximum r/min* available within M42 gear range. Atypical range may be 4000 r/min or even higher.

With the preset maximum spindle speed limit of 1500 r/min (G50 S1500), the spindle will be constantly increasing its speed, but only *until* it reaches the 1500 preset r/min, then it will *remain at that speed* for the rest of cut.

At the control, CNC operator can easily change the maximum limit value, to reflect true setup conditions or to optimize cutting values.

Spindle speed is preset - or *clamped* - to the maximum *r/min* setting, by programming the S function together with the G50 preparatory command. If the S function is in a block not containing G50, the control will interpret it as a new spindle speed (CSS or r/min), that will be active from that block on. *This error may be very costly!*

Use caution when presetting maximum r/min of the spindle!

Maximum spindle speed can be clamped in a separate block or in a block that also includes the current tool coordinate setting. Typically, the combined setting is useful at the beginning of a tool, the separate block setting is useful if the need arises to change the maximum spindle speed in the middle of a tool, for instance, between facing and turning cuts using the same tool.

To program G50 command as a separate block, anywhere in the program, just issue the preparatory command combined with the spindle speed preset value. Such a block will have *no effect* whatsoever on any active coordinate setting, it represents just another meaning of G50 command. The following examples are all correct applications of G50 command for both, the old-style coordinate setting and/or the maximum spindle speed preset:

N12 G50 X20.0 Z3.0 S1500 *Double meaning*

N38 G50 S1250 *Single meaning*

N15 G50 X8.5 Z2.5 *Single meaning*

N40 G50 Z4.75 S700 *Double meaning*

If the CNC lathe supports old G92 instead of G50, keep in mind that they have exactly the same meaning and purpose. On old controls, G50 command is more common than G92 command but programming method is the same.

Part Diameter Calculation in CSS

Often, knowing at *what diameter* the spindle will actually be clamped can be a very useful information. Such knowledge may influence preset value of spindle speed clamping. To find out at what diameter the *Constant Surface Speed* will remain fixed, the formula that finds the r/min at a given diameter must be reversed:

D	12 ft / min

$$\frac{D}{r / \min}$$

+ where . . .

D = Diameter where CSS stops (inches)

12 = Multiplying factor - feet to inches conversion

ft/min = Active surface speed - feet per minute

= Constant 3.1415927

r/min = Preset maximum spindle speed

Example - Imperial units:

If the preset value in a program is G50 S1000 and the surface speed is set as G96 S350, the CSS will be clamped when it reaches the 1.3369 inches:

$$D = (12 \ 350) / (1000)$$

$$= 1.3369015$$

$$= 1.3369$$

The formula may be shortened, as before:

$$D = \frac{3.82 \text{ ft / min}}{r / \min}$$

Formulas based on the imperial system, can be adapted to a metric environment:

$$D = \frac{1000 \text{ m / min}}{r / \min}$$

+ where . . .

D = Diameter where CSS stops (in mm)

1000 = Multiplying factor - meters to mm conversion m/min = Active surface speed - meters per minute

= Constant 3.1415927

r/min = Preset maximum spindle speed

Just like in its imperial version counterpart, metric formula may be shortened as well:

$$D = \frac{3183. \text{ m / min}}{r / \min}$$

Example - Metric units:

If the preset value in a program is G50 S1200 and the surface speed is set as G96 S165, the CSS will be clamped when it reaches the 43.768 mm:

$$D = (1000 \ 165) / (\ 1200)$$

$$= 43.767609 = 43.768 \text{ mm}$$

Cutting Speed Calculation

Constant Surface Speed (CSS) - cutting speed - is required for virtually all turning and boring operations on a CNC lathe. It is also the basic source of cutting data, from which the spindle speed is calculated for virtually all machining center operations. Now, consider a very common scenario:

CNC operator has optimized the current cutting conditions, including the spindle speed, so they are very favorable. Can these conditions be applied to subsequent jobs?

Yes, they can - and should - provided that certain critical requirements will be satisfied:

Machine and part setup are equivalent

Cutting tools are equivalent

Material conditions are equivalent

Other common conditions are satisfied

If these requirements are met, the most important source data is the spindle speed actually used during machining. Once the spindle speed is known, the cutting speed (CSS) can be calculated and used for any other tool diameter, providing the requirements above are met.

In a nutshell, the whole subject can be quickly summed up by categorizing it as a *cutting speed* calculation - calculation of *Constant Surface Speed (CSS)*, when the tool or part diameter *and* the spindle speed are known.

From there on, it is a simple matter of formulas:

To calculate *Cutting Speed* in imperial units:

	D r / min
ft / min	_____
	12

EXAMPLE:

A 5/8-inch drill works very well at 756 r/min - what is its cutting speed in ft/min?

$$\text{ft/min} = (3.14 \ 0.625 \ 756) / 12 = 123.64$$

To calculate *Cutting Speed* in metric units:

	D r / min
m / min	_____
	1000

EXAMPLE:

A 7 mm end mill works very well at 1850 r/min - what is its cutting speed in m/min?

$$\text{m/min} = (3.14 \ 7 \ 1850) / 1000 = 40.66$$

The major benefit of using this method is a significant reduction of time spent at the CNC machine, usually required to find and 'fine-tune' the optimum spindle speed during setup or part optimization. Knowing when a particular cutting speed can be applied is one of several optimizing methods available to CNC programmers.

12. FEED RATE CONTROL

Feed rate is the closest programming companion to spindle function. While the spindle function controls spindle speed and spindle rotation direction, feed rate controls *how fast the tool will move*, usually to remove excessive material (stock). In this digital book, subject of *rapid positioning*, sometimes called a *rapid motion* or *rapid traverse motion*, is not considered a true feed rate and will be described separately, starting on page 147.

A. FEED RATE CONTROL

Cutting feed rate is the speed at which the cutting tool removes the material by cutting action

A cutting action may be a rotary tool motion (drilling and milling, for example), rotary motion of the part (lathe operations), or other action (flame cutting, laser cutting, water jet, electric discharge etc.). Feed rate function is used in CNC program to select correct feed rate amount, suitable for the desired action.

Two feed rate types are used in CNC programming:

Feed rate per minute - mm/min or in/min

Feed rate per revolution - mm/rev or in/rev

The most common types of CNC machines, machining centers and lathes, can be programmed in either feed rate mode. In practice, it is much more common to use the *feed rate per minute* on machining centers and the *feed rate per revolution* on lathes.

There is a significant difference in G-codes used for machining centers and lathes.

FEED RATE	Milling	Turning Group A	Turning Group B	Turning Group C
Per minute	G94	G98	G94	G94
Per revolution	G95	G99	G95	G95

G-code *Group A* is the most commonly used on Fanuc controls and also in this publication

Another type of special feed rate is called the *inverse time feed rate*. It is used in some rotary applications only.

B. FEED RATE FUNCTION

Calling address for a feed rate word in a program is the address *F*, followed by a number of digits. The number of digits following address *F* depends on the feed rate mode selection and machine tool application. Decimal place is normally allowed for feed rate programming.

Feed rate per Minute

For milling applications, all cutting feed rate in linear and circular interpolation modes is programmed in *millimeters per minute (mm/min)* or in *inches per minute (in/min)*. The amount of feed rate is the distance a cutting tool will travel in one minute (60 seconds). This value is modal and is canceled only by another *F*-address word. The main advantage of feed rate per minute is that it is *not* dependent on current spindle speed. That makes it very useful in milling operations, using a large variety of tool diameters. Standard abbreviations for feed rate per minute are:

Millimeters per minute ... mm/min

Inches per minute ... in/min (or older ipm / IPM)

The most typical format for feed rate per minute is F5.1 for the metric system and F5.1 for the

imperial system.

For example, feed rate of 15.5 inches per minute, will be programmed as F15.5. In metric system, feed rate amount of 250 mm/min will appear in the program as F250.0. A slightly different programming format may be expected for special machine designs.

One important item to remember about feed rate is the *range* of available feed rate values. Feed rate range of the control system *always exceeds* that of machine servo system. For example, feed rate range of a CNC machine with Fanuc CNC system is typically described in metric units, for example, 30000 mm/min may be the maximum for a machine with 60000 mm/min rapid motion. Minimum cutting feed rate depends on the machine builder - it could be 0.1 mm/min or 0.1 in/min or a totally different amount.

In milling, programming command (G-code) for *feed rate per minute* is G94. For most machines, it is set automatically, by the system default and does not have to be written in the program. For lathe operations, feed rate per minute is used very seldom. In *Group A*, G-code for feed rate per minute is G98, for *Groups B and C* it is G94. CNC lathes use primarily *feed rate per revolution* mode.

Feed rate per Revolution

For CNC lathe work, feed rate is not measured in terms of time, but as the actual distance the tool travels in one spindle revolution (rotation). This *feed rate per revolution* is common on lathes (G99 for Group A). Its value is modal, and another feed rate function cancels it (usually the G98). Lathes can also be programmed in *feed rate per minute* mode (G98), to control feed rate when the spindle is stationary. Two standard abbreviations are used for *feed rate per revolution*:

Inches per revolution in/rev (or older ipr)

Millimeters per revolution mm/rev

The most programming typical format for feed rate per revolution is *three* decimal places in *metric* system, and *four* decimal places in *imperial* system. Metric feed rate example of 0.42937 mm/rev will be programmed as F0.429 on most controls. In imperial units, a feed rate of 0.083333 in/rev will be applied in the CNC program as F0.0833 on most controls. Many modern control systems accept feed rate of up to five decimal places for metric units and six decimals for imperial units.

Be careful when rounding feed rate values. For turning and boring operation, reasonably rounded feed rates are quite sufficient. Only in tapping and single point threading the feed rate precision is critical for a proper thread lead, particularly for long or very fine threads. Some Fanuc controls can be programmed with up to six decimal places feed rate precision for threading only.

Programming command for feed rate per revolution is G99. For most lathes, this is the system default, so it does not have to be written in the program, unless the opposite command G98 is also used.

It is relatively more common to program a *feed rate per minute* (G98) for a CNC lathe program, than it is to program a *feed rate per revolution* (G99) in a milling program. The reason is that on a CNC lathe, this command controls feed rate while the spindle is *not* rotating. For example, during a bar feed operation, a part stopper is used to 'push' the bar to a precise position in the chuck or a collet, or a pull-put finger to 'pull' the bar out. Rapid feed would be too fast and feed rate per revolution is not applicable. Feed rate per minute is used instead. In cases like these, G98 and G99 commands are used in a lathe program as required. Both commands are modal, and one cancels the other.

C. FEED RATE SELECTION

To select the best feed rate, one that is most suitable for a given job, some general knowledge of machining is useful. This is an important part of programming process and should be done

carefully. A feed rate selection depends on many factors, most notably on:

Spindle speed - in r/min - revolutions per minute

Tool diameter [M] or tool nose radius [T]

Surface requirements of the part

Cutting tool geometry

Machining forces

Part setup

Tool overhang (extension)

Cutting length motion

Amount of material removal (depth or width of cut)

Method of milling (climb or conventional)

Number of flutes in material (for milling cutters)

Safety considerations

The last item is safety - always a programming responsibility number one, to assure the safety of people and equipment. Safe speeds and feeds are only two aspects of much wider safety awareness in CNC programming.

D. ACCELERATION AND DECELERATION

During a typical contouring operation, the direction of cutting motion is changed quite often. There is nothing unusual about it, with all the intersections, tangency points and clearances. In contouring, it means that in order to program a sharp corner on a part, the tool motion along X-axis in one block will have to change into a motion along Y-axis in the next block. To make the change from one cutting motion to another, control must *stop the X-motion* first, then *start the Y-motion*. Since it is impossible to *start* at a full feed rate instantly, without *acceleration*, and equally impossible to *stop* feed rate without *deceleration*, a possible cutting error may occur. This error may cause sharp corners on the part contour to be cut with an undesirable overshoot, particularly during very high feed rates or extremely narrow angles. It only occurs during a cutting motion in G01 mode, as well as in G02 and G03 modes, but *not* in rapid motion mode G00. During rapid motion, motion deceleration is automatic - and away from the part.

In routine CNC machining, there is a small chance of ever encountering such an error. Even if the error is present, it will likely be well within tolerances.

If the error does need correction, Fanuc controls provide two commands that can be used to correct the problem:

G09	Exact stop	(One block only)
G61	Exact stop mode	(modal)

Exact stops increase the cycle time. For programs used on older machines, they may be required in some cases.

FEED RATE CONTROL

Exact Stop Command

The first of two commands that control the feed rate when machining around corners is G09 command - *Exact Stop*. This is an *unmodal* command and has to be repeated in every block, whenever it is required.

In program example O1301, there is no provision for acceleration and deceleration. That may cause uneven corners, due to the rather high feed rate of F90.0 (in/min):

O1301 (NORMAL CUTTING) ...

N13 G00 X15.0 Y12.0

N14 G01 X19.0 F90.0

N15 Y16.0

N16 X15.0

N17 Y12.0 ...

By adding G09 exact stop command in the program, the motion in that block will be fully completed *before* the motion in the other axis will start - O1302:

O1302 (G09 CUTTING) ...

N13 G00 X15.0 Y12.0

N14 G09 G01 X19.0 F90.0

N15 G09 Y16.0

N16 G09 X15.0 N17 Y12.0 ...

Example O1302 guarantees a sharp corner at all three positions of the part. If only one corner is critical for sharpness, program the G09 command in the block that terminates at that corner - O1303:

O1303 (G09 CUTTING) ...

N13 G00 X15.0 Y12.0

N14 G01 X19.0 F90.0

N15 G09 Y16.0

N16 X15.0

N17 Y12.0 ...

G09 command is useful only if a handful of blocks require deceleration for a sharp corner. For a program where all corners must be precise, constant repetition of G09 is not very efficient.

Exact Stop Mode Command

The second command that corrects an error at sharp corners is G61 - *Exact Stop Mode*. It is much more efficient than G09 and functions identically. The major difference is that G61 is a *modal* command that remains in effect until it is canceled by G64 cutting mode command. G61 shortens programming time, but not the cycle time. It is most useful when G09 would be repeated too many times in the same program, making it unnecessarily too long - O1304.

O1304 (G61 CUTTING) ...

N13 G00 X15.0 Y12.0

N14 G61 G01 X19.0 F90.0

N15 Y16.0

N16 X15.0

N17 Y12.0 N18 G64 ...

Note that program example O1304 is identical in results to program O1301. In both cases, exact stop check applies to *all* cutting motions - unmodally in program O1301, modally in program

O1304. Also note an additional block N18. It uses G64 command - *normal cutting mode*. Normal cutting mode is the default setting when machine power is turned on and is not usually programmed. *Figure 13-1* illustrates tool motion *with* and *without* G09/G61 command. The large overshoot amount shown is exaggerated only for the illustration, in reality it is very small.

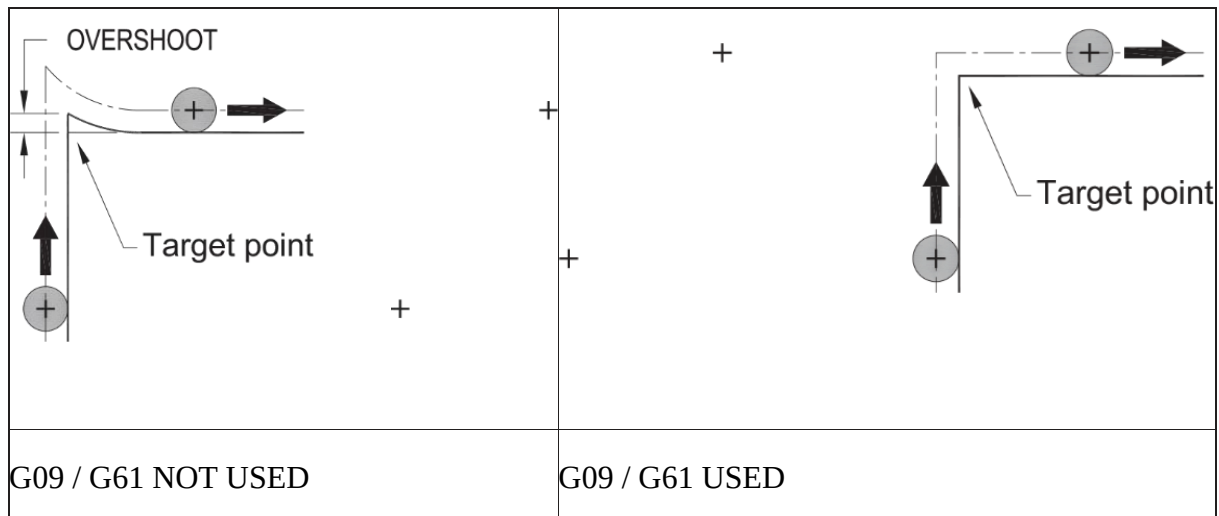


Figure 13-1

Feed rate control around corner - Exact Stop commands the overshoot is exaggerated for clarity

Automatic Corner Override

While a cutter radius offset is in effect for milling cutters, feed rate at the contour change points is normally *not* overridden. In a case like this, the preparatory command G62 can be used to automatically override cutting feed rate at the corners of a part. This command is active until G61 command (exact stop check mode), G63 command (tapping mode), or G64 command (cutting mode) is programmed.

Tapping Mode

Programming in tapping mode G63 will cause the control system to ignore any settings of *feed rate override switch*, except the 100% setting. It will also cancel function of the Feed hold key, located on control panel. Tapping mode will be canceled by programming G61 command (exact stop check), or G62 command (automatic corner override mode selection), or G64 command (cutting mode selection).

Cutting Mode

When cutting mode G64 is programmed or is active by the system default, it represents *normal* cutting mode. When this command is active, exact stop check G61 will *not* be performed, neither will automatic corner override G62 or tapping mode G63. That means motion acceleration and deceleration will be done normally (as per control system) and feed rate overrides *will be* effective. This is the most common default mode for a typical control system.

Cutting mode can be canceled by programming the following commands: G61 (exact stop mode), G62 (automatic corner override mode) or G63 (tapping mode).

G64 command is not usually programmed, unless one or more of other feed rate modes are used in the same program. To better understand G62 and G64 modes, compare illustrations in *Figure 13-2*.

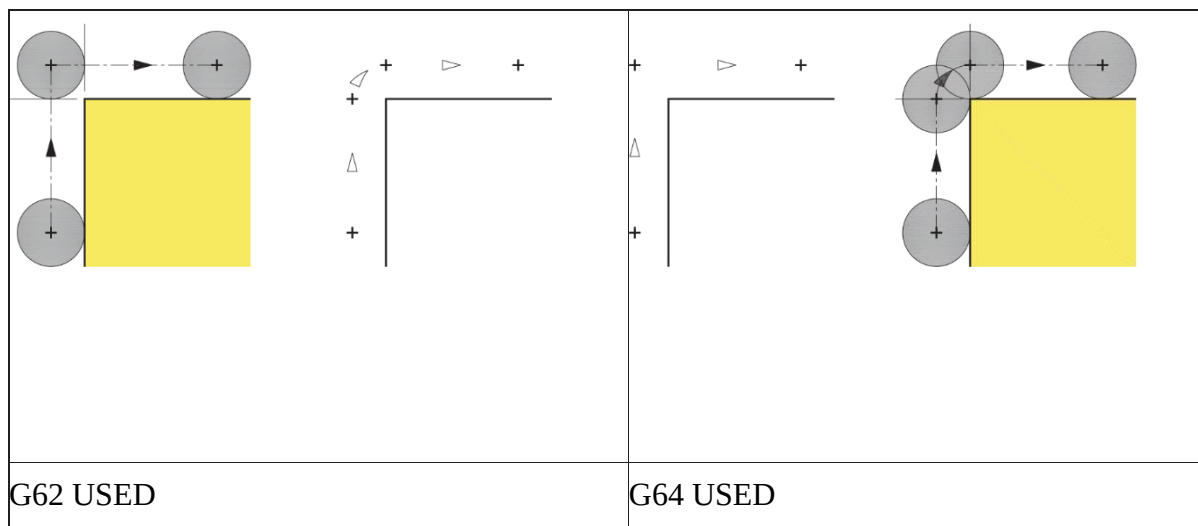


Figure 13-2

Corner override mode G62 and default G64 cutting mode

E. CONSTANT FEED RATE

Beginning on *page 243*, the main topic is *circular interpolation*. In this chapter are detailed explanations and examples of maintaining a constant cutting feed rate for inside and outside arcs, from the *practical* point of view. At this point, the main focus is on actual *understanding* of the constant feed rate, rather than its *application*.

In programming, normal process is to calculate coordinate values for all contour change points, based on the part *drawing*. Cutter radius that produces the *centerline* of a tool path is typically disregarded. When programming arcs to drawing dimensions, rather than to the cutter centerline, feed rate applied to the programmed arc always relates to the *programmed* radius, *not* the actual radius cut at the cutter center.

When cutter radius offset is active and the arc tool path is offset by cutter radius, the *actual* arc radius that is cut can be either *smaller* or *larger*, depending on the preset offset amount for cutting tool motion.

It is very important to understand that *effective* cutting radius will *decrease* in size for all internal arcs, and it will *increase* in size for all external arcs. Since the cutting feed rate does not change automatically during cutter radius offset mode, *it must be adjusted* in the program. Usually, this adjustment is not necessary, except in cases where the surface finish is of great importance or cutter radius is very large. This consideration applies only to circular motions, not to linear cutting.

Circular Motion Feed rates

Setting feed rates for circular motions is generally the same as for linear feed rates. In fact, most programs do not change feed rate for linear and circular tool motions. If part surface finish is important, so called 'normal' feed rate must be adjusted *higher* or *lower*, with consideration of the cutter radius, the type of radius cutting (outside or inside arc) and cutting conditions. The larger the cutter radius, the more reason the cutting feed rate for programmed arcs will need some correction.

In case of arc cutting, the equidistant tool path - after applying cutter radius offset - may be *much larger* or *much smaller* than the arc programmed to drawing dimensions.

Feed rate for compensated arc motions is always based on the currently programmed linear motion feed rate. Look for a more detailed explanation on *page 253*, with an illustration and examples. First, here is the standard formula for calculating a linear feed rate:

F_l	$r / \min \quad F_t \quad n$	

+ where . . .

F_l = Linear feed rate (in/min or mm/min) $r/\min$ = Spindle speed (revolutions per minute) F_t = Feed rate per tooth (cutting edge) n = Number of cutting edges (flutes or inserts)

Based on the linear feed rate formula, arc feed rate adjustments are influenced by the *side* of machined arc - *outside* or *inside* arc. Linear feed rate should be *increased* for outside arcs and *decreased* for inside arcs.

For *outside* arcs, the feed rate is generally adjusted *upwards*, to a higher value:

	$F (R \ r)$	
F_o	$\frac{\quad}{R}$	

+ where . . .

F_o = Feed rate for outside arc

F_l = Linear feed rate (normal feed rate for the tool) R = Outside radius of part r = Cutter radius

FEED RATE CONTROL

For *inside* arcs, the feed rate is generally adjusted *downwards* to a lower value:

	$F (R \ r)$	
F_i	$\frac{\quad}{R}$	

+ where . . .

F_i = Feed rate for inside arc

F_l = Linear feed rate (normal feed rate for the tool) R = Inside radius of part r = Cutter radius

F. MAXIMUM FEED RATE

Maximum programmable cutting feed rate for CNC machines is determined by the machine manufacturer, not the control manufacturer. For example, maximum cutting feed rate on a particular machine may be 10000 mm/min, although the CNC system can support a feed rate several times greater. This is applicable to all controls, but there are additional programming considerations for CNC lathes, where *feed rate per revolution* is the main method of programming motion velocity of a cutting tool.

Maximum Feed rate Considerations

Maximum *cutting* feed rate per revolution is always restricted by the *programmed spindle speed* ($r/\min$) and *maximum rapid traverse rate* of a CNC lathe. It is quite easy to program feed rate per revolution too high without even realizing it. This problem is most common in single point threading, where cutting feed rates can be unusually high.

ACNC machine cannot deliver heavier feed rates than the maximum amount it was designed for - threading results would be unacceptable. When unusually heavy feed rates and fast spindle speeds are combined in the *same* program, it is advisable to check whether the active feed rate does not exceed the maximum feed rate allowed on a given machine. It can be calculated as the *maximum feed rate per revolution*, according to the following formula:

$$F_{\max} = \frac{R_{\max}}{r / \min}$$

+ where . . .

$F_{\max}$ = Maximum feed rate per revolution (units/rev)

$R_{\max}$ = Lower of the maximum feed rate, selected between X and Z axes

$r / \min$ = Spindle speed in revolutions per minute

$R_{\max}$ is set either in *mm/min* or *in/min*, depending on the input units selected. Page 359 covers more details relating to feed rate limits for threading.

G. FEED HOLD AND OVERRIDE

While running a program, the programmed feed rate may be temporarily suspended or changed by using one of two available features of the control system. One is called a *Feed hold* switch, the other is a *feed rate override* switch. Both switches are standard and allow the CNC operator to manually control programmed feed rate during program execution. They are located on the machine operation panel.

Feed hold Switch

Feed hold is a push button that can be toggled between *Feed hold ON* and *Feed hold OFF* modes. It can be used for both feed rate modes, *feed rate per minute* or *feed rate per revolution*. On many controls, Feed hold will stop not only a cutting feed with G01, G02, G03 in effect - it will also stop rapid motion G00. Other program functions will remain active during a Feed hold state.

For certain machining operations, Feed hold function is *automatically disabled* and becomes ineffective. This is typical for tapping and threading, using G84 and G74 tapping cycles on machining centers and threading operations using the G32, G92 and G76 on CNC lathes.

Feed rate Override Switch

Feed rate override is normally controlled by means of a special rotary switch, located on the control panel of the CNC unit - Figure 13-3.

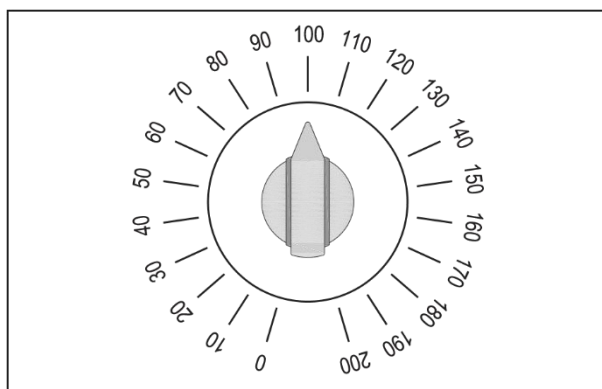


Figure 13-3

Typical feed rate override switch

This rotary switch has marked settings or divisions, indicating the *percentage of programmed*

feed rate. A typical range of a feed rate override is 0 to 200% (or 0 to 150%), where 0 may be no motion at all or the slowest motion, depending on the machine design. 200% setting *doubles* all programmed feed rates. Programmed feed rate of 12.0 in/min (F12.0) is 100% feed rate. If the override switch is set to 80%, the actual cutting feed rate will be 9.6 in/min. If the switch is set to 110%, the actual cutting feed rate will be 13.2 in/min.

This simple logic applies to metric system as well. If the programmed feed rate is 300 mm/min, it becomes 100%. An 80% feed rate override results in 240 mm/min cutting feed rate and a 110% feed rate override setting is equivalent to 330 mm/min for the cutting tool.

Feed rate override switch works equally well for lathes and *feed rate per revolution*. For example, the programmed feed rate of 0.014 in/rev will result in the actual feed rate of 0.0126 in/rev with 90% feed rate override and 0.0182 in/rev with 130% feed rate override. If a very precise feed rate per spindle revolution is required, be careful with override settings. For example, a programmed feed rate is F0.012, in *inches per revolution*. A change by a single division on the override dial will either *increase* or *decrease* the programmed amount by a full 10 percent. Therefore, the feed rate will be 0.0108 at 90%, 0.0120 at 100%, 0.0132 at 110%, etc. In most cases, such precise feed rate is not required, but keep in mind that some feed rates will not be accessible, for example, a feed rate of 0.0115 in/rev, because of the fixed 10% increments on the override switch.

In single point threading mode G32, the feed rate override switch is *disabled*. Feed rate override is also disabled for tapping cycles G84 and G74 on machining centers, and for single point threading cycles G92 and G76 on lathes. If the tapping mode is used for milling systems, with the command G63, both *feed rate override* and *Feed hold* functions are disabled - *through the program!*

Control system offers two feed rate override functions for cutting motions *other than* tapping or threading cycles. They are M48 and M49. These are *programmable* functions but may not be available for all controls.

Feed rate Override Functions

Although the feed rate function uses address F, two special miscellaneous functions M can be used in a program to set the *feed rate override ON* or *OFF*. On the machine operation panel, a switch is provided for feed rate override. If the CNC operator decides that programmed feed rate has to be temporarily *increased* or *decreased*, this switch is very handy. On the other hand, during machining operations, where the cutting feed rate *must* be used as programmed, the override switch has to be set to 100% only, not to any other setting.

A good example are special tapping operations *without* cycles, using G01 and G00 preparatory commands. Functions M48 and M49 are used precisely for such purposes:

M48	Feed rate override cancel function is OFF, which means feed rate override is <i>active</i>
M49	Feed rate override cancel function is ON, which means feed rate override is <i>inactive</i>

M48 function enables the CNC operator to use the feed rate override switch freely; M49 function will cause feed rates to be executed *as programmed*, regardless of the feed rate override switch setting on the machine panel. The most common usage of these two functions is for tapping or threading *without* a cycle, where the exact programmed feed rate *must* always be maintained. The following example shows a typical programming method:

N10 S500 M03 (USING TAP 12 TPI)

...

N14 G00 X5.0 Y4.0 M08

N15 Z0.25

N16 M49 (DISABLE FEED RATE OVERRIDE)

N17 G01 Z-0.625 F41.0 M05

N18 Z0.25 M04

N19 M48 (ENABLE FEED RATE OVERRIDE)

N20 G00 X.. Y.. M05 N21 M03 ...

The tapping motion occurs between blocks N16 and N19 and the feed rate override is disabled for these blocks.

H. E ADDRESS IN THREADING

Some CNC lathes with older controls (like Fanuc 6T) use feed rate address *E* for threading, rather than the more common address *F*.

Feed rate function *E* is similar to the *F* function. It also specifies the thread lead as *feed rate per revolution*, in *mm/rev* or in *in/rev*, but it has a *greater decimal place accuracy*. On older Fanuc control system model 6T, for example, the range for thread lead is:

English - Fanuc 6T control:

F = 0.0001 to 50.0000 in/rev E = 0.000001 to 50.000000 in/rev Metric - Fanuc 6T control:

F = 0.001 to 500.000 mm/rev

E = 0.0001 to 500.0000 mm/rev

On the newest control models, FS-0T to Fanuc 30T, the ranges are similar (there is no *E* address), but the safest way to find the available ranges is to lookup the specifications for your control system.

The *E* address is redundant on all newer controls and is retained only for compatibility with older programs that may be used on machines equipped with newer controls. The available threading feed rate ranges vary between different control systems and depends on the type of feed screw and the input units used in the program. Virtually all controls now allow *F*-address with extended precision, when applied to threading.

13. TOOL FUNCTION

Each numerically controlled machine using an automatic tool changer must have a special tool function (T-function) that can be used in the program. This function controls the behavior of the *cutting tool*, depending on the type of machine. There are noticeable differences between T-functions used on CNC machining centers and those used on CNC lathes. There are also differences between similar controls for the same machine type. The normal programming address for the tool function uses the address T.

For CNC machining centers, T-function typically controls the *tool number* only. For CNC lathes, the function controls indexing to the *tool station number*, as well as the *tool offset number*.

A. T-FUNCTION

So far, the tool function was covered as it applied to CNC machining centers. CNC lathes also use the tool function T, but with a completely different structure.

Lathe Tool Station

A typical slant bed lathe uses a polygonal turret holding all external and internal cutting tools in special pockets or *stations*. These *tool stations* are similar to a tool magazine on machining centers. Typically, their design accepts 8, 10, 12 or more cutting tools - *Figure 14-7*.

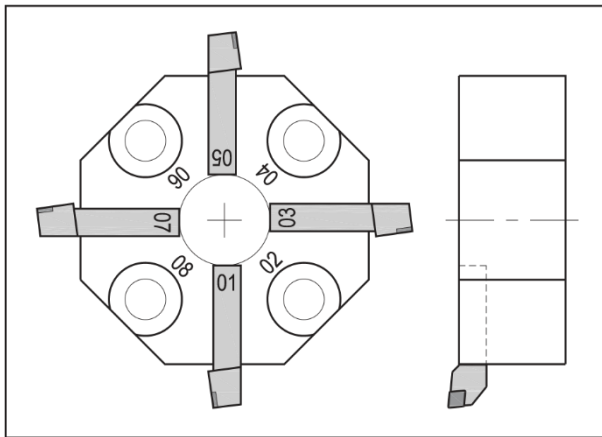


Figure 14-7

Typical view of an octagonal lathe turret (tools 1 to 8)

Some CNC lathe models start adopting the tool changer type similar to machining centers, with many more tools available and away from the work area.

Since all tools are normally held in a single turret, the one selected for cutting will always *carry along* all other tools into the work area. This may be a design whose time has passed but it is still very commonly used in industry. Because of a possible interference between a tool and the machine or part, care must be taken not only of the *active* cutting tool, but also of *all other tools* mounted in the turret, while considering all possible collision situations.

Tool Indexing

To program a tool change, or rather to *index* a cutting tool into its active position, the T-function must be programmed using a proper format. For CNC lathe, this format calls for the address T, followed by *four* digits - *Figure 14-8*.

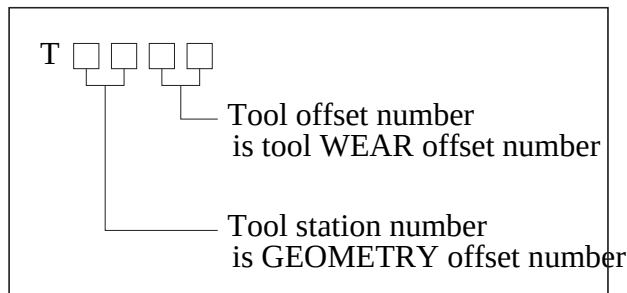


Figure 14

Structure of a 4-digit tool number for CNC lathes

It is important to understand this function well. Think about the four digits as *two pairs* of (two) digits, rather than four single digits. Leading zeros within each pair may be omitted. Each pair has its own meaning:

The *first* pair (first and second digits), control the tool index station and the geometry offset.

Example:

T01xx - selects the tool mounted in position one and activates geometry offset number one

The *second* pair (third and fourth digits), control the tool wear offset number used with the selected tool.

Example:

Txx01 - selects the wear offset register number one

It is customary, but not mandatory, to match both pairs, if possible. For example, tool function T0101 will select tool station number one, geometry offset number one and the associated tool wear offset register number one. This format is easy to remember and should be used every time, if only one offset number is assigned to the tool number.

If *two or more different wear offsets* are used for the same tool, it is not possible to match both pairs. In such a case, *two or more different* wear offset numbers must be programmed and applied to *the same* tool station number:

Example:

T0101 for turret station 01, geometry offset 01 and wear offset 01 Example:

T0111 for turret station 01, geometry offset 01 and wear offset 11

The first pair is always the tool station number *and* the geometry offset number. The latter example assumed that tool wear offset 11 is not used by another tool. If tool 11 is used with offset 11, another suitable wear offset number must be selected, for example 21, and program it as T0121. Most controls have 32 or more offset registers for geometry offsets and another 32+ wear offsets registers.

All offset values are applied to CNC lathe program by first registering their values into *offset registers*.

B. TOOL OFFSET REGISTERS

The word *offset* has been mentioned already several times with two adjectives - with the expression *geometry offset* and the expression *wear offset*. What exactly is an *offset*? What is the difference between one offset and the other?

On the OFFSET display of a typical Fanuc control, there is a choice of two screens, both very similar in appearance. One is called the *Geometry Offset* screen, the other is called the *Wear Offset* screen. Figure 14-9 and Figure 14-10 show examples of both screens, with typical (i.e.,

reasonable) sample entries (imperial units' input).

OFFSET - GEOMETRY				
No.	X-offset	Z-offset	Radius	Tip
01	-8.4290	-16.4820	0.0313	3
02	-8.4570	-14.7690	0.0000	0
03	-8.4063	-16.3960	0.0156	3
04	-8.4570	-12.6280	0.0000	0
05	-8.4350	-16.4127	0.0000	0
06	-9.8260	-13.2135	0.0313	2
07	0.0000	0.0000	0.0000	0

Figure 14-9

Example of the GEOMETRY offset screen display

OFFSET - WEAR				
No.	X-offset	Z-offset	Radius	Tip
01	0.0000	0.0000	0.0313	3
02	0.0000	0.0150	0.0000	0
03	0.0036	0.0000	0.0156	3
04	0.0000	-0.0250	0.0000	0
05	0.0010	-0.0022	0.0000	0
06	-0.0013	0.0000	0.0313	2
07	0.0000	0.0000	0.0000	0

Figure 14-10

Example of the WEAR offset screen display

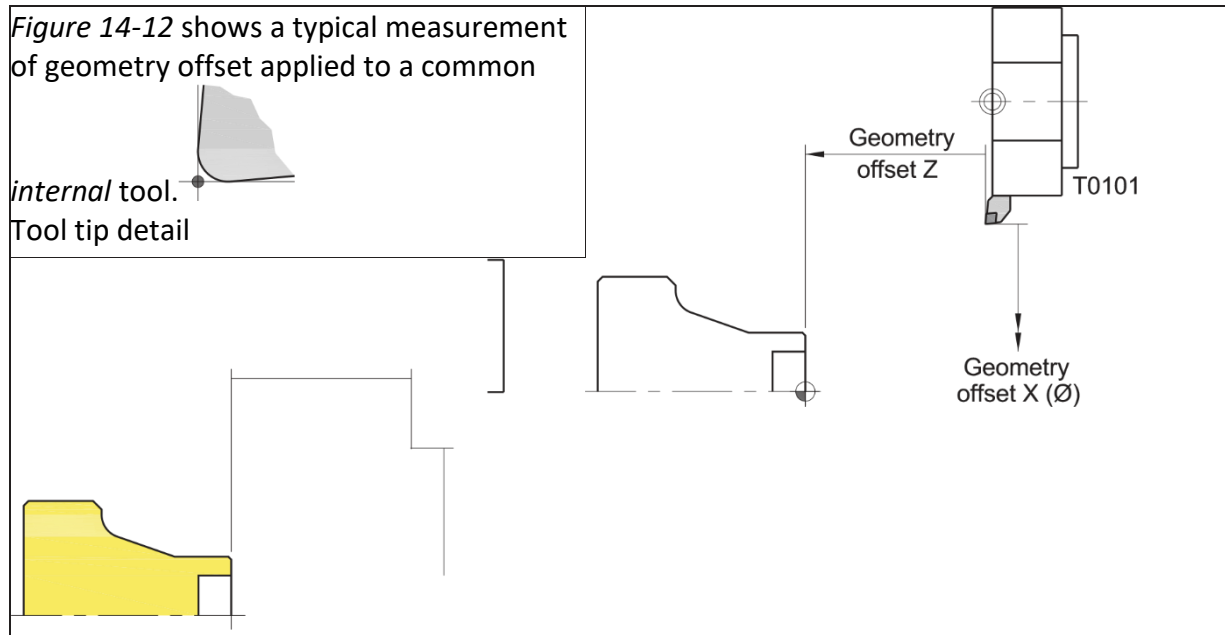
Geometry Offset

Geometry offset number *always* matches the turret station number selected. Machine operator measures and enters geometry offsets for all tools used in the program.

GEOMETRY offset amount is always measured from the machine zero position

Distance from machine zero position will reflect the distance from tool reference point to part reference point (program zero). *Figure 14-11* shows a typical measurement of geometry offset applied to a common *external* tool.

All X settings will normally have *diameter* values and are stored as *negative* for a typical rear lathe of a slant bed type. Z-axis settings will normally be also negative (positive values are possible but impractical). How to actually measure geometry offset is a subject of CNC machine tool operation training, not programming. Manual and automatic methods are available for this purpose.

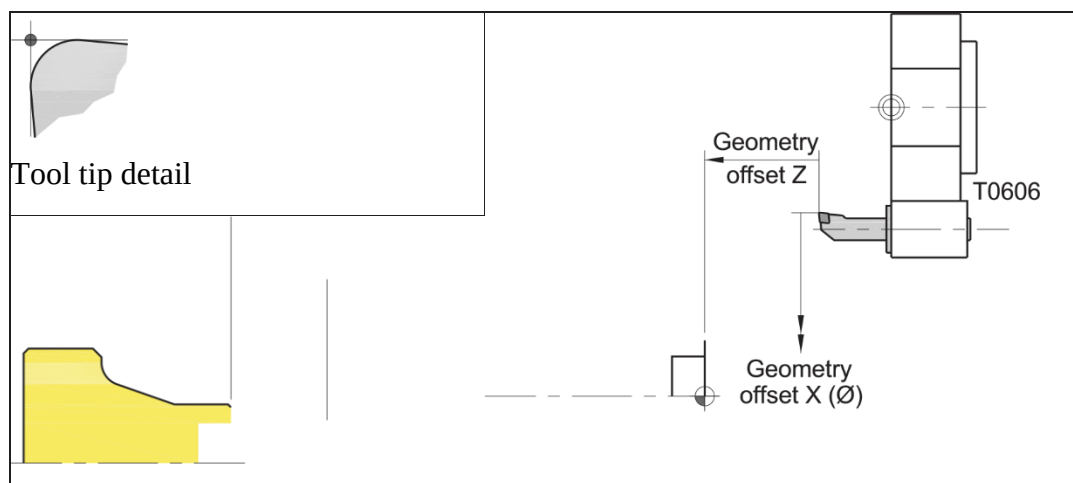


One last possibility relating to geometry offset is illustrated in Figure 14-13. It shows geometry offset applied to any tool used on the spindle center line (always at X0 position). These tools include center drills, drills, taps, reamers, etc. Their X-offset setting will always be the same.

Wear Offset

In CNC program development, many dimensions used will be those in the part drawing. For example, a dimension of 3.0000 inches, is programmed as X3.0. This figure does not reflect any implied dimensional tolerances. Program entries X3.0, X3.00, X3.000 and X3.0000 have *exactly* the same result. What is needed to maintain dimensional tolerances, particularly when they are tight? What has to be done with a worn-out tool that is still good enough to cut a few more parts? The answer is that the programmed tool

Figure 14-11 path must be adjusted, *fine-tuned*, to match the machining conditions. Program itself will not be changed, but a *wear offset* for the selected tool is applied.



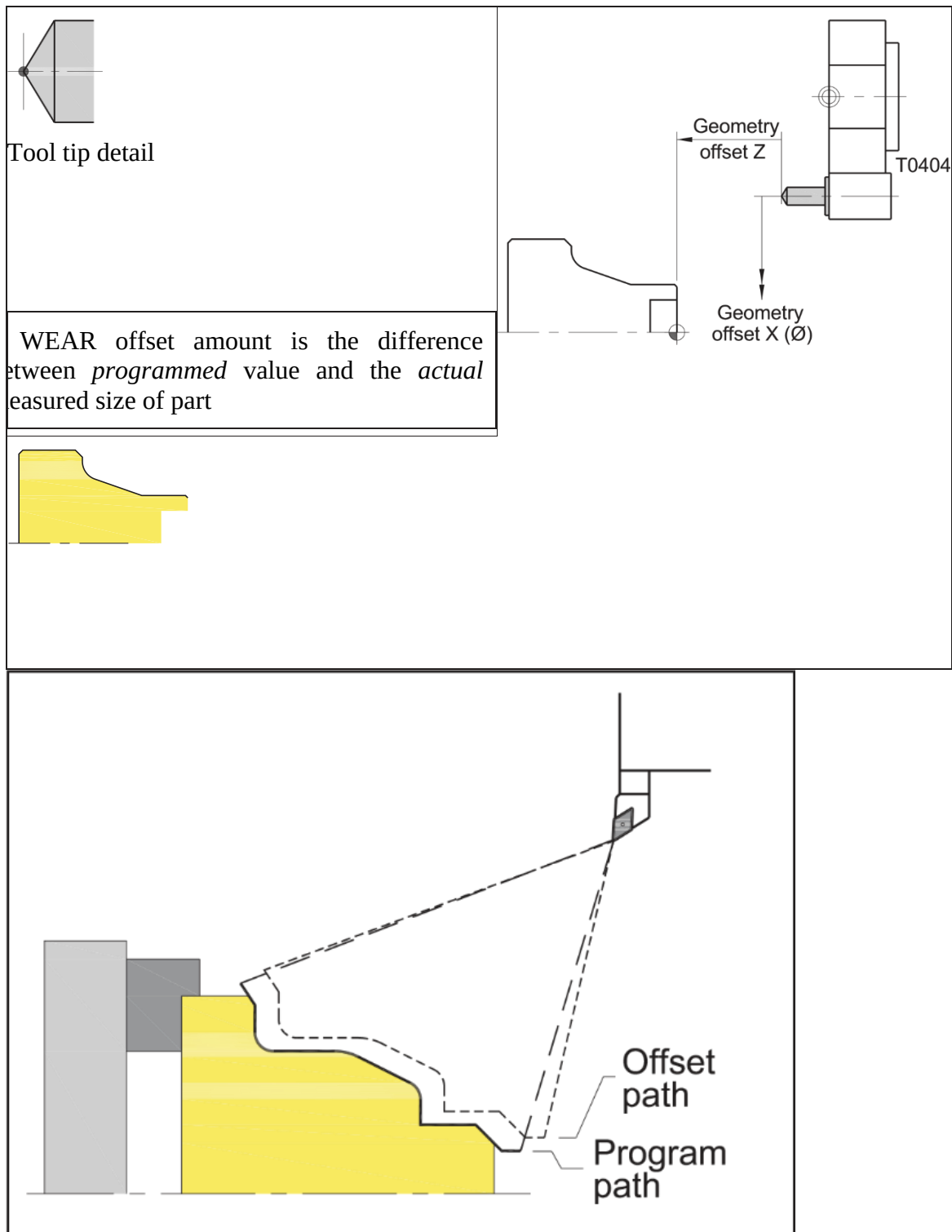


Figure 14-14

Figure 14-14 illustrates the principle of tool wear offset, although the scale is exaggerated for emphasis.

Figure 14-12

Typical geometry offset for internal (boring) tools

Programmed tool path and tool path with wear offset

Actual wear offset setting has only one purpose - it compensates between the programmed value, for example that of 3.0 inch, and the actual dimension *as measured* during inspection, for example, 3.004. The differential value of -0.004 is entered into the wear offset register. This is the

offset number specified as *the second pair* of tool function used in the program. Since the program uses diameters for the X-axis, offsets will also be applied to diameter. Details on this kind of adjustment are more useful to CNC machine operator, but any part programmer will benefit from them

Figure 14-13 as well.

Typical geometry offset for center line (drilling) tools

Wear Offset Adjustment

To illustrate the concept of wear offset adjustment on a rear type of lathe, T0404 in the program will be used as an example. The goal is to achieve an outside diameter of 3.0 inches and tolerance of 0.0005. Initial setting of the wear offset in register *Txx04* will be zero. Relevant section of the program may look something like this:

```
N31 M01
...
N32 T0400 M42
N33 G96 S450 M03
N34 G00 G42 X3.0 Z0.1 T0404 M08
N35 G01 Z-1.5 F0.012 N36 ...
```

When the machined part is inspected (measured), it can have only one of three possible inspection results:

On-size dimension

Oversize dimension

Undersize dimension

If the part is measured *on size*, there is no need to interfere. Both tool setup and program are working correctly. If the part is *oversize*, it can usually be recut, if machining an *outside* diameter. For an *inside diameter*, the exact opposite will apply. Recut may damage the surface finish, which should be a concern. If the part is *undersize*, it becomes a scrap. The aim is to prevent all *subsequent* parts from being undersize as well. The following table shows inspection results for all three existing possibilities:

Measurement	External diameter	Internal diameter
ON size	Size OK	Size OK
OVER size	Recut possible	SCRAP
UNDER size	SCRAP	Recut possible

Let's go a little further. Whether the part will be oversized or undersized, something has to be done to prevent this problem from happening again. Common action to take is *adjusting the wear offset amount*. Again, the emphasis here is that this is an example of an *outside* diameter.

External diameter X3.0 in the example may result in 3.004 diameter measured size. That means it is 0.004 oversize - on diameter. CNC operator, who is in charge of offset adjustments, will change the currently set 0.0000 value in the X-register of the wear offset 04 to -0.0040. The subsequent cut should result in a part that will be measured within specified tolerances.

If the part in the example is undersize, say at 2.9990 inches, the wear offset must be adjusted by +0.0010 in the X-positive direction. Measured part is a scrap.

Principle of the wear offset adjustment is logical. If the machined diameter is *larger* than the drawing dimension allows, the wear offset is changed into the *minus* direction, towards the spindle center line, and vice versa. This principle applies equally to external and internal diameters. The only practical difference is that an oversized external diameter and undersize internal diameter *can* be recut (see the table above). Several practical examples of using the wear offset creatively are described on *page 306*.

The R and T Settings

The last items are the R and T columns (*Geometry and Wear*). Both offset screen columns are only useful during setup. The *R column* is the radius column, *T column* is the tool tip orientation column (*Figure 14-15*).

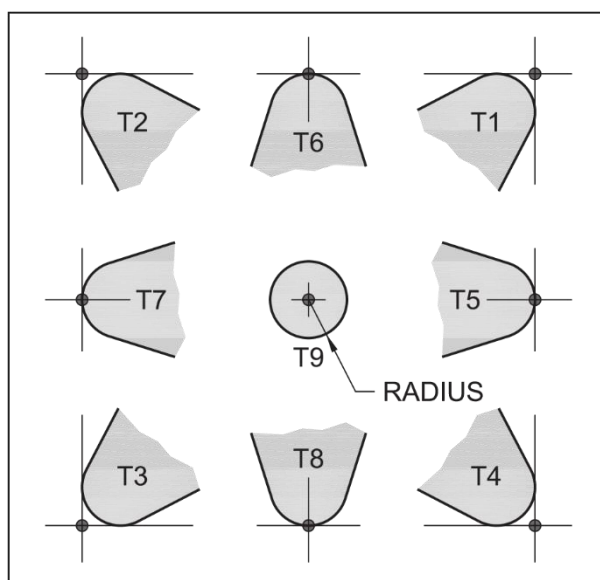


Figure 14-15

Arbitrary tool tip orientation numbers used with tool nose radius compensation (G41 or G42 mode)

The main rule of using R and T columns is that they are only effective in tool nose radius offset mode. If no G41 or G42 is programmed, settings in these two columns are irrelevant. If G41/G42 command is used, non-zero values for that tool must be set in both columns. The R-column requires *tool nose radius* of the cutting tool, the T-column requires *tool tip orientation number* of the cutting tool. Both are described on *page 275* in more detail. The most common tool nose radiuses for turning and boring are:

- 1/64 of an inch = 0.0156 or 0,4 mm
- 1/32 of an inch = 0.0313 or 0,8 mm
- 3/64 of an inch = 0.0469 or 1,2 mm

Tool tip numbers are arbitrary and indicate the tool orientation number used to calculate the nose radius offset, regardless of tool setting in the turret.

14. REFERENCE POINTS

In the previous chapters, the basic relationship between machine geometry and part setup was covered. CNC programmers work in a fairly precise environment, and several mathematical relationships are of extreme importance.

There are three major environments in programming that require an established mathematical relationship:

Environment	Relationship consists of ...
Machine	Machine tool + Control system (CNC unit)
Part	Workpiece + Drawing + Material
Tool	Holder + Cutting tool

Each environment by itself is independent of the other two. If the relationship is not apparent right away, consider the *sources* of each environment:

MACHINE TOOL is made by a company specializing in machine tools, usually not controls or cutting tools

... *this environment is combined with* ...

CONTROL SYSTEM is made by a company specializing in the application of electronics to machine tools. They do not normally manufacture machine tools or cutting tools

PART (workpiece) is a unique engineering design developed in a company that does not manufacture machine tools, control systems, or cutting tools and holders

CUTTING TOOLS are a specialty of tooling companies, which may or may not make cutting tool holders. These companies do not manufacture machine tools or CNC systems

These sources inevitably converge when customer buys a CNC machine. A certain engineering design (part), must be machined on a machine tool from one manufacturer, using control system of another manufacturer, cutting tools from yet another manufacturer, and tool holders from a fourth source. These sources are similar to a musical quartet of first-class musicians who never played together. In both cases there is a need to create perfect harmony.

By itself, each environment is not very useful. A machine without tools will not yield any profit; a tool that cannot be used on any machine is not going to benefit the manufacturing either. A part cannot be machined without tools.

The common point here is that all three environments cannot be useful without some 'teamwork'. They have to work together - *they have to interact*.

For programming purposes, these relationships and interactions are based on one common denominator of *each* environment - *a reference point*.

Reference point is a fixed or selected arbitrary location on the machine, on the tool and on the part. A fixed reference point is a precise location along two or more axes, designed during manufacturing or setup. Some reference points are established by the programmer, during programming process. In these three environments, three reference points are needed - one reference point for each of the available groups:

Machine reference point .. Machine zero or Home

Part reference point .. Program zero or Part zero

Tool reference point .. Tool tip or Command point

In a typical machine shop language, these reference points have somewhat more practical meaning. *Home position* or a *machine zero* are synonymous terms for *machine reference point*. *Program zero*, or *part zero*, or *part origin* are terms commonly used instead of the more official term *part reference point*. And the names *tool tip* and *tool command point* are commonly used for the *tool reference point*.

A. REFERENCE POINT GROUPS

The first group covers the *CNC machine tool* or *CNC machine* for short, which is the combination of machine proper and the control system. Numeric values that relate to the CNC machine tool include a variety of dimensions, specifications, parameters, ranges, ratings, etc. When a part is set in a fixture on the machine table or mounted into a lathe chuck, collet, face plate, or other work holding device, there is a second group of numbers to consider. They are *part considerations*, such as its size, its height, diameter, shape, etc., that are unique to each job. Finally, the third group of numbers relate to the *cutting tools*. Each cutting tool has its individual features, as well as features that are shared with many other cutting tools.

All available numeric values have a meaning - they are not merely numbers - they are actual *values* or *setting amounts* that programmers and operators have to work with individually as well as together.

Reference Point Groups Relationship

The key to any successful CNC program is to make all three groups to work in a coordinated way. This goal can only be achieved by understanding the principles of reference points and how they work. Each reference point can have one of two characteristics:

Fixed reference point

Flexible or floating reference point

A fixed reference point is set by the machine manufacturer as part of the hardware design and cannot be physically changed by the user. A CNC machine has at least one fixed reference point. When it comes to deciding the reference points for the part or the cutting tool, CNC programmer has certain degree of freedom. Part reference point (program zero) is always a flexible point, meaning its actual position is in programmer's hands. Reference point for the mounted cutting tool can be either fixed or flexible, depending on machine design.

B. MACHINE REFERENCE POINT

Machine zero point, often called *machine zero*, *home position* or just *machine reference position*, is the origin of machine coordinate system. The location of this point may vary between machine manufacturers, but the most obvious differences can be found between individual machine types, namely vertical and horizontal models.

In general terms, a CNC machine has two, three, or more axes, depending on the type and model. Each axis has a maximum range of travel that is fixed by the manufacturer. This range is usually different for each axis. If the CNC operator exceeds the range on either end, an error condition known as *overtravel* will occur. Not a serious problem, but one that could be annoying. During machine setup, particularly after the power has been turned on, the position of all axes has to be preset to be always the same, from day to day, from one part to another. On older machines, this procedure is done by setting a grid, on modern machines, by performing a *machine zero return command*. Fanuc and many other control systems prevent automatic operation of a machine tool, unless the machine zero return command has been performed at least once - when the power to the machine has been turned on. A welcome safety features.

On all CNC machines that use typical coordinate system, machine zero is located at the *positive* end of each axis travel range. For a typical three-axis vertical machining center, look at the part in the XY plane, that is straight down from the tool position (tool tip). Also look into the XZ plane

(operator's front view of the machine), or into the YZ plane (operator's right-side view of the machine). These three planes are perpendicular to each other and together create so called *work cube* or *workspace* - Figure 15-1.

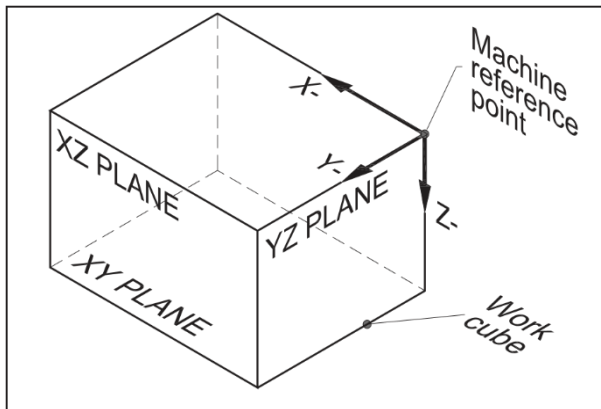


Figure 15-1

Machine reference point and axes orientation for a vertical machine

The cubical shape shown is useful only for overall understanding of machine work area. For programming and setup, the majority of work is done with one or two axes at a time. To understand the work area and machine zero point in a plane, look at the machine from the top (XY machine plane) and from the front (XZ machine plane). Figures 15-2 and 15-3 illustrate both views.

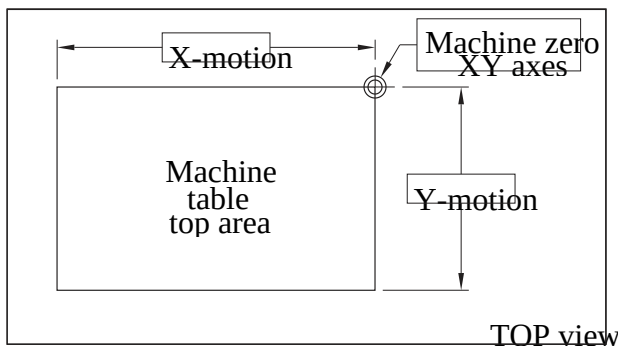


Figure 15-2

Top view of a vertical machine as viewed towards the table

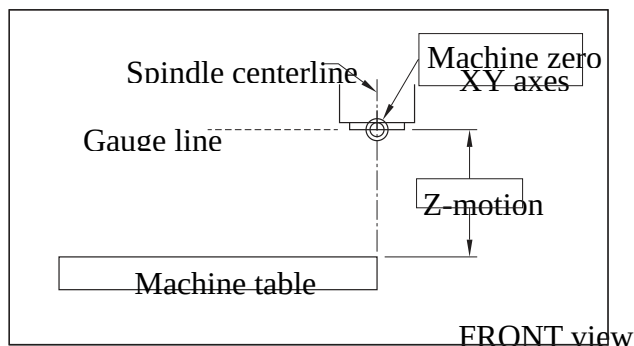


Figure 15-3

Front view of a vertical machine as viewed from the front

Compare the two views. In top view, the upper right corner is also the spindle center line shown in the front view.

Also note that in front view, there is a dashed line identified as the *gauge line*. This is an imaginary location for the proper fit of tool holder tapered body and is set by the machine manufacturer. Inside of the spindle is a precision machined taper that accepts tool holder with the

cutting tool. Any tool holder mounted in the spindle will be in exactly the same position. The Z-motion illustrated will be shortened by the cutting tool projection. This subject of tool referencing is discussed later in this chapter.

Return to Machine Zero

In manual mode, the CNC operator physically moves selected axes to machine zero position. The operator is also responsible to register this position into the control system, if necessary (older systems only). Never turn off power to the machine, while machine slides are *at* or very *close to* machine zero position. Being too close will make manual machine zero return more difficult later, after the power had been restored. A clearance of 1.0 inch (25.0 mm) or more for each axis *from* machine zero is usually sufficient. A typical procedure to physically reach the machine zero position will follow these steps:

- Turn the power on (machine and control)
- Select machine zero return mode
- Select the first axis to move (usually Z-axis)
- Repeat for all other axes
- Check the lighted *in-position* indicators
- Check the position screen display
- Set display to zero, if necessary

Mainly for safety reasons, the first selected axis should be the Z-axis for machining centers and the X-axis for lathes. In both cases, either axis will be moving *away* from work, into a clear area. When the axis has reached machine zero position, a small indicator light on the control panel turns on to confirm that the axis actually reached machine zero. The machine is now at its reference position, at machine zero, or at the machine reference point, or at *home* - whichever term is used in the shop. Indicator light ON is the confirmation for each axis. Although the machine is ready for use, a good operator will go one step further. On the *position* display screen, the actual relative position should be set to zero readout for each axis, as a standard practice, if it is not set to zero automatically by the control. Typically, the POS button on the control panel selects position screen display.

C. PART REFERENCE POINT

A part ready for machining is located within the machine motion limits. Every part must be mounted in a device that is safe, suitable for the required operation and does not change position for any other part of the job run. Fixed location of the device is very important for consistent results and precision. It is also very important to guarantee that each part of the job is set the same way as the first part. Once the setup is established, part reference point can be selected.

This vital reference point will be used in a program to establish the relationship with machine reference point, reference point of the cutting tool and the drawing dimensions.

Part reference point is commonly known as *program zero* or *part zero*. Because the coordinate point that represents program zero can be selected by the programmer almost anywhere, it is not a fixed point, but a *floating* point. As this point is selectable, more details can be covered - after all, it is the programmer who selects part zero.

Program Zero Selection

When selecting program zero, often in the comfort of programmer's office, a major decision is made that will influence the efficiency of part setup and its machining in the shop. Always be very attentive to all factors that are for and against program zero selection in a certain position.

In theory, the program zero point may be selected literally anywhere. That is not much of some

advice, although true in mathematical terms. Within practical restrictions of the machine operations, only the most advantageous possibilities should be considered. Three such considerations should govern the selection of program zero: **Accuracy of machining**

Convenience of setup and operation

Safety of working conditions

Machining Accuracy

Machining accuracy is paramount - all parts must be machined exactly to the same drawing specifications. Accuracy is also important consideration in repeatability. All parts in the batch must be the same and all subsequent jobs must be the same as well.

Convenience of Setup and Operation

Operating and setup convenience can only be considered once the machining accuracy is assured. Working easier is everyone's desire. An experienced CNC programmer will always think of the effect the program has in machine shop. Defining program zero that is difficult to set on the machine or difficult to check is not very convenient. It slows down the setup process even more.

Working Safety

Safety is always important to whatever we do - machine and part setup are no different. Program zero selection has a lot to do with safety of the machining operation.

We look at the typical considerations of program zero selection for vertical machining centers and lathes individually. Differences in part design influence the program zero selections as well.

Program Zero - Lathes

On CNC lathes, program zero selection is simple. There are only two axes to consider - the vertical X-axis and the horizontal Z-axis. Because of lathe design, the X-axis program zero selection is always the spindle center line.

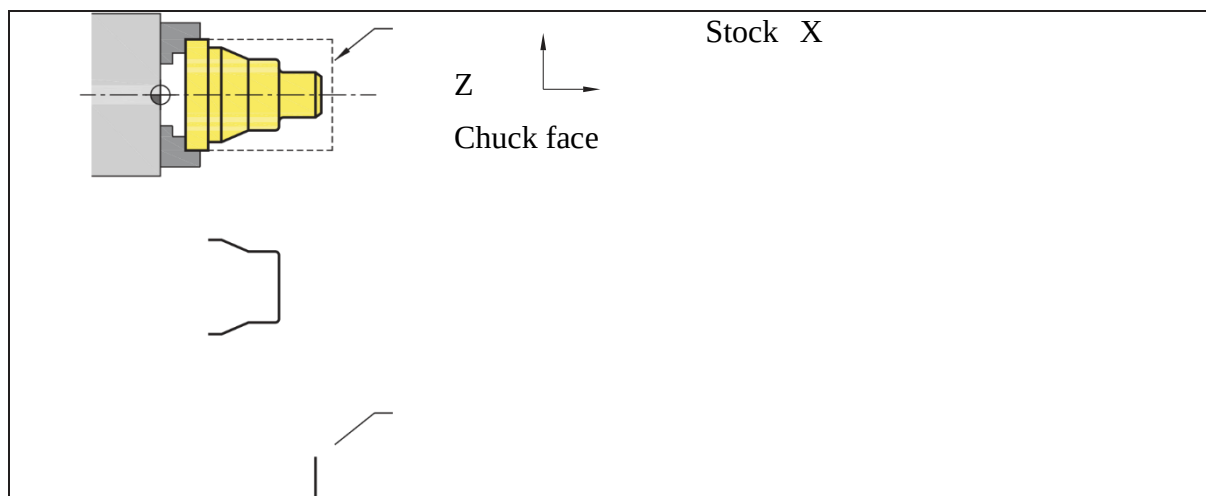
On CNC lathes, program zero for the X-axis *MUST* be on the center line of the spindle

For the Z-axis, three popular methods are used:

Chuck face . . . main face of the chuck

Jaw face . . . locating face of the jaws

Part face . . . front of the finished part



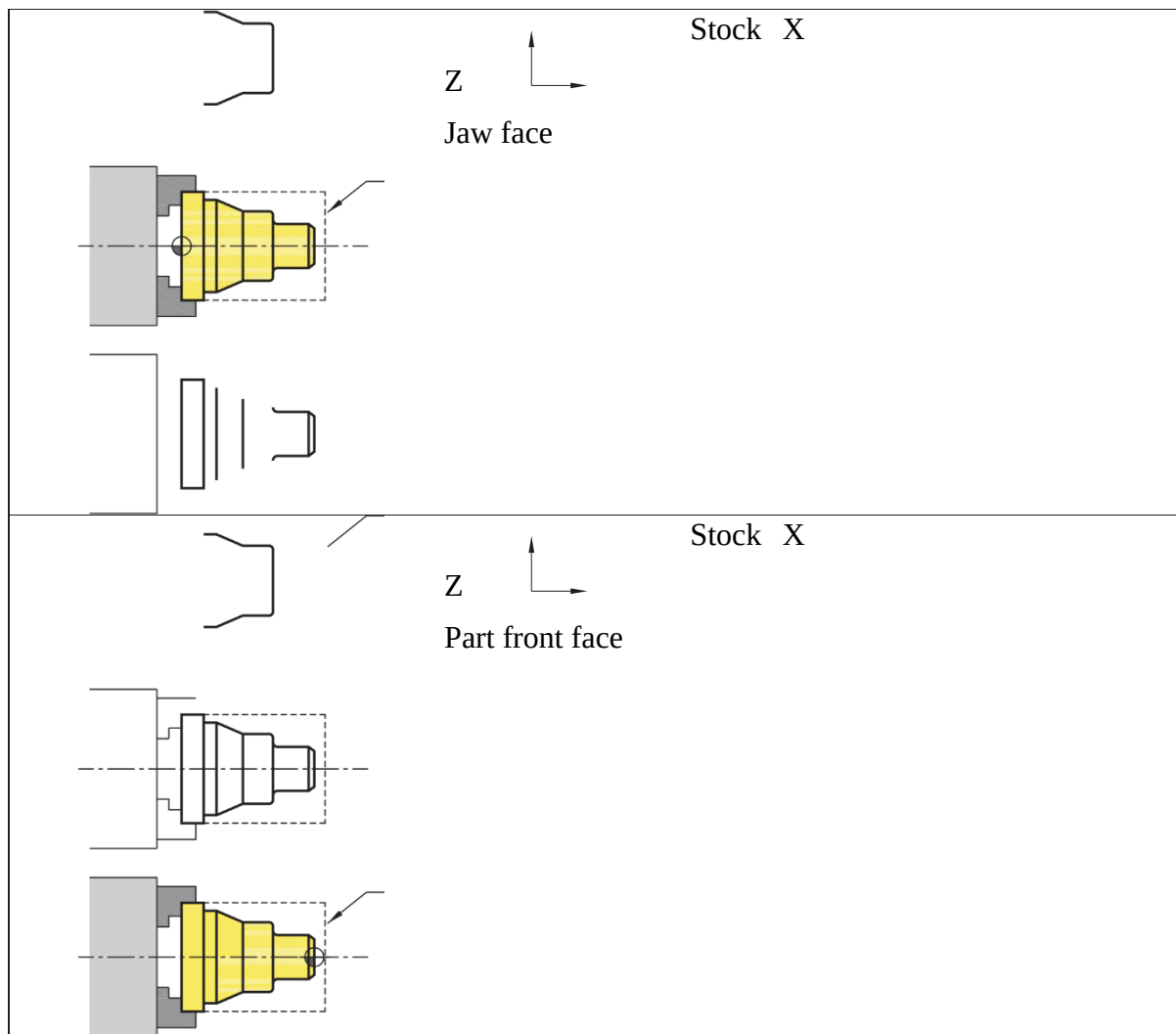


Figure 15-10

Common program zero options for a CNC lathe - center line is X0

Figure 15-10 illustrates the three options. In practical setup, a chuck face offers only one benefit - it can be easily touched with the tool edge, using feelers to prevent tool chipping. On a negative side, unless the part rests against chuck face, additional calculations are needed for the coordinate data and drawing dimensions cannot be used easily.

Jaw or fixture face presents more favorable situation. The face can also be touched with the tool and is consistent for all parts. This location may benefit machining irregular shapes, such as castings, forgings and similar parts.

Many lathe parts require machining at both ends. During the first operation, material stock for the second operation must always be added to every Z value. This is the main reason why CNC programmers keep away from program zero located on jaw or fixture face, except in special cases.

The most popular method is setting program zero on the *front face* of the *finished* part. This is not a perfect selection either but has many other advantages. The only disadvantage is that during setup, there is often no finished face. Many operators add the width of the rough face to the setup or cut a small face for the tool to touch.

What are the benefits of program zero at the front face? One benefit is that many drawing dimensions along Z-axis can be transferred directly into the program, normally with a *negative* value. A lot depends on the dimensioning method, but in majority of cases, CNC programmer benefits. Another benefit, probably the most important, is that a negative Z-value of a tool motion indicates the work area, a positive Z-value is in the clear area. During program development, it is easy to forget a minus sign for the Z-axis cutting motions. Such an error, if not caught in time, will

position the tool *away* from the part, with tailstock as a possible obstacle. It is a wrong position, but a better one than hitting the part. Examples in this digital book use program zero at the *front finished face*, unless otherwise specified.

D. TOOL REFERENCE POINT

The last reference point is related to the tool itself. In milling and related operations, the tool reference point is usually the intersection of tool centerline and the lowest positioned cutting tip (edge).

In turning and boring, the most common tool reference point is an imaginary tool point of the cutting insert, because most tools have a cutting edge with a built-in radius.

For tools such as drills and other point-to-point tools used in milling or turning, the reference point is always the extreme tip of the tool, as measured along Z-axis. *Figure 15-11* shows some common tool tip points.

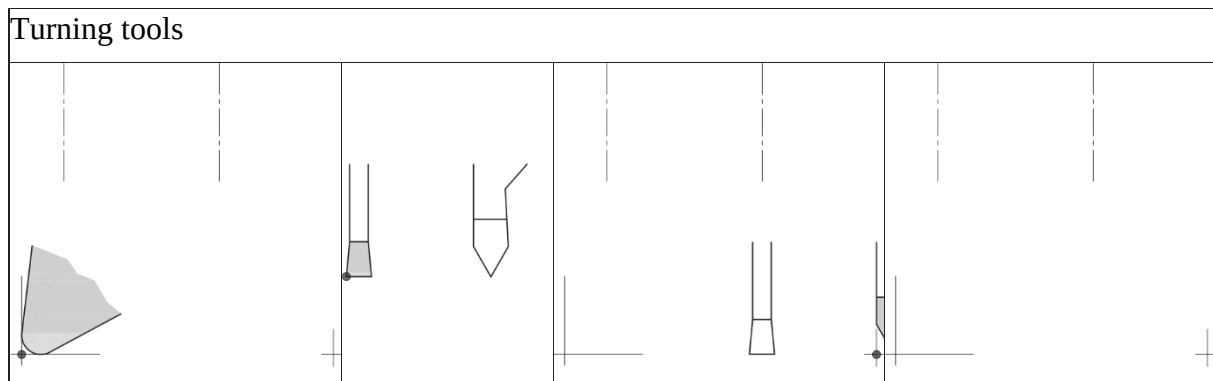


Figure 15-11

Typical tool reference points for various cutting tools

All three reference point groups are related. An error in one setting will have an adverse effect on another setting. Knowledge of reference points is important to understand register commands, offsets and machine geometry.

15. REGISTER COMMANDS

The three reference points available for CNC programming must be harmonized to work together correctly. Having the reference points for the *part* (i.e., program zero) and the cutting *tool* (i.e., tool tip) available, there has to be some means to associate them together, to fit them together. There must be some means to 'tell' the control system exactly where each tool is physically located, within the machine work area, *before* it can be used. The oldest method to do all this is to *register* the current tool position into the control system memory, through the program. This method required a command called the *Position Register*.

A. POSITION REGISTER COMMAND

Preparatory command for the tool position register is G92 for machining centers and G50 for (most) lathes:

G92	Position register command	(Used in milling)
G50	Position register command	(Used in turning)

Some CNC lathes also use the G92 command, but lathes supplied with Fanuc and similar controls normally use G50 command instead. In practical applications, both G92 and G50 commands have identical meaning, and the following discussion applies to both commands equally. In the first part of this chapter, the focus will be on milling applications using G92 command, lathe applications using G50 command will be explained later.

In modern CNC programming, both position register commands were replaced by a much more sophisticated and flexible feature called the *Work Offsets* (G54 to G59), (*see page 127 for details*) and the *Tool Length Offset* (G43), (*see page 135 for details*). However, there are still quite a few older machine tools in shops that do not have the luxury of the G54 series of commands. There are also many companies using programs developed years ago, but still running on modern CNC equipment. In those cases, understanding the position registration command is an essential skill. This command has always been one that some programmers and operators found a little difficult to understand. In reality, it is a very simple command.

First, a look at some more detailed definition of this command. A typical description only specifies *Position Register Command*, which by itself is not very conclusive.

Position Register Definition

A little more verbose definition of the position register command could be expressed this way:

Position register command sets the tool location as the distance and direction FROM ... the program zero,

TO ... the tool current position, measured along the axes

Note that this definition does not mention *machine zero* at all - instead, it mentions the *current tool position*. This is a very important distinction. Current tool position *may be*

at machine zero, but it also *may be* somewhere else, within travel limits of the machine axes.

Also note the emphasis on *from-to* direction. By definition, the distance is unidirectional, between the program zero and the current tool location. The direction is always *from* program zero, *to* tool position, never reversed. In a program, the correct identification of each axis value (positive, negative, or zero) is always required.

Position register is only applicable in absolute mode of programming, while G90 command is in effect. It has no use in the incremental programming mode G91. In practical programming, virtually all programs written in incremental mode do begin in absolute mode, in order to reach the first tool location.

Programming Format

As the command name suggests, tool position data associated with G92 command will be *registered* (i.e., stored) into the control system memory.

Format for G92 command is as follows:

G92 X.. Y.. Z..

In all cases, the address of each axis specifies the distance *from program zero to tool reference point* (tool tip). CNC programmer provides all coordinates based on the program reference point (program zero), discussed earlier. Any additional axis will also have to be registered with G92, for example the B-axis for the indexing table on horizontal machining centers.

Tool Position Setting

The only purpose of G92 command is to *register the current tool position into the control memory* - nothing more!

No machine motion will ever occur in a block containing the G92 command!

The effect of G92 can be seen on the *absolute position* screen display. At all times, absolute position display has some values for each axis, including zero. When G92 command is executed, all current values of the display will be *replaced* with the settings specified with G92. If an axis was not specified with G92, there will be no change of display for that axis. At the machine, the operator has a major responsibility - to match the actual tool setting with the settings specified in the G92 command.

B. LATHE APPLICATION

For the CNC lathes with Fanuc and similar controls, G50 command is used instead of G92 command:

G50 X.. Z..

If G92 is used for a lathe, the command is similar:

G92 X.. Z..

Either command has exactly the same definition and rules as for milling - it indicates the distance *from program zero, to the current tool position* along axes.

No machine motion will occur in a block containing G50 or G92 command!

Commands G50 and G92 are identical, except that they belong to two different G-code groups. Fanuc actually offers three G-code groups for lathe controls. Based on history, typical Japanese made controls use G50, whereby typical US made controls used G92. A cooperative US and Japanese venture known as *GE Fanuc* (*General Electric* and *Fanuc*) produces controls that are the most common in North American industry and using the G50 command.

To program position register for lathe applications is very similar to that of G92 for mills. However, due to design of CNC lathes, where all tools are mounted in the turret, the *projection* of each tool (for both axes) from the turret holder must be considered. Not only that, but possible interference must also be prevented, because all mounted inactive tools move simultaneously along the one that is used for cutting. In milling, all non-active tools are safely out of way, placed in a tool magazine. Several new designs of CNC lathes are available, where tool changer on the lathe resembles the milling type.

Tool Setup

The most important programming decision for lathe work relates to the setup. Although there are several options to select from, some are preferable to others.

Probably the most practical approach for lathe setup will be to have the tool change position for

all tools corresponding to the machine zero position. This is a very easy position to move the turret to, just using the control panel switches. The position register measured to machine zero does have one major disadvantage - it may be *too far* for most jobs, particularly on larger lathes along the Z-axis. Just imagine a tool motion of 30 inches or more along the Z-axis only to index the turret and then the same 30-inch motion back to continue a cutting cycle. It is not efficient at all. There is a solution, however.

Much more efficient method is to select the tool indexing position for all tools as *close to the part* as possible. This position should always be based on the *longest* tool mounted in the turret (usually internal tools), whether such tool is used in the program or not. If there is enough clearance for the longest tool in the turret, there will also be enough clearance for all remaining shorter tools.

A possible compromise of the two methods described is to keep tool indexing position at the X-axis machine zero only (which is usually not too distant) and just establish the Z-axis position.

On a CNC lathe, do not forget to keep in mind the general layout of all tools in the turret, to prevent a collision with the part, the chuck, or the machine.

There is other, but less common, methods to setup a tool on the lathe using the G50 command.

Three-Tool Setup Groups

On a typical slant bed CNC lathe, equipped with a polygonal turret (6 to 14 stations), all cutting tools reside in individual stations of the turret. During tool indexing, only the selected tool is in the active station. Upon evaluation of the type of tools used for CNC lathe operations, it will be clear that there are only three groups of cutting tools, based on the type of work they normally do:

Tools working on the part center line

Tools working externally on the part

Tools working internally on the part

If the position register for each group is understood well, it will be easy to apply it to any tool within a group, regardless of the number of tools used.

Center Line Tools Setup

Lathe tools classified as center line tools are typically center drills, spot drills, standard twist drills, indexable carbide drills, taps, reamers, and so on. Even an end mill can be used at the spindle center line. All tools in this group have a single common denominator, whereby the tool tip is always located on the spindle center line (X0), while they cut. These tools must always be setup exactly at 90 to the work face (parallel to the Z-axis).

Position register value in X-axis is from the spindle center line (X0) to the center line of the tool. For the Z-axis, the position register value is measured from program zero to the tool tip. Typically, center line tools will have a fairly large overhang - that means their G50 value along the Z-axis will be relatively small, when compared to external tools, which generally do not extend that much.

Figure 16-4 illustrates a typical setup for center line tools, using an indexable drill as an example.

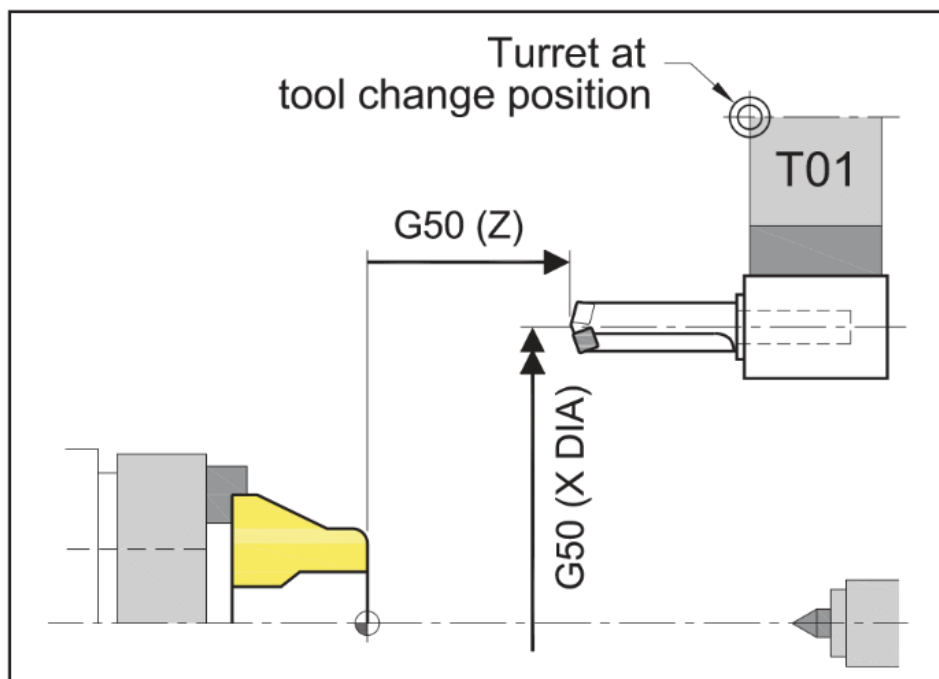


Figure 16-4

Typical G50 setting for center line lathe tools

REGISTER COMMANDS

External Tools Setup

For external machining operations such as roughing and finishing outside diameters, taper cutting, grooving, knurling, single point threading, part-off and others, the cutting tool is rather small and approaches the part in an open space with generous clearances.

Position register value is measured from the program zero to the imaginary tool tip of the insert (see details at the end of this chapter). In case of tools like threading tool or a grooving tool, G50 amount is usually measured to the left side of the insert, for safety reasons.

Figure 16-5 illustrates a typical position register setup for an external tool (turning tool shown in the example).

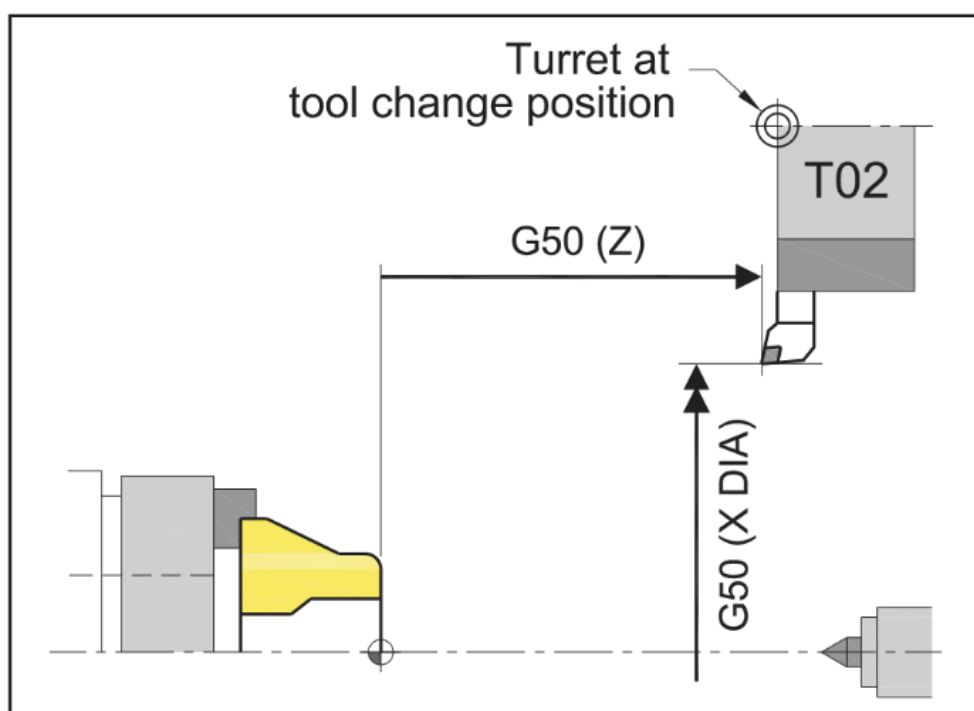


Figure 16-5

Typical G50 setting for external lathe tools

Internal Tools Setup

Internal tools are all tools that do majority of their work inside of a part, in a pre-machined hole, core or other cavity. Typically, we may first think of a boring bar, but other tools can be used as well for various internal operations. For example, an internal grooving and internal threading are also common operations on CNC lathes. Setup rules for Z-axis apply in the same way for internal tools as for external tools of the same type.

Along the X-axis, tool position register setting must be made to the imaginary tip of the insert. Figure 16-6 shows a typical position register setup for an internal tool (boring bar shown in the example).

All three illustrations (Figures 16-4, 16-5 and 16-6) also show a possible order of the three operations (drill - turn bore) for a typical CNC lathe job. Note that the turret position is identified as a *tool change position*, not necessarily as the machine zero position. That means G50 may be set anywhere within travel limits of the machine, even at machine zero.

121

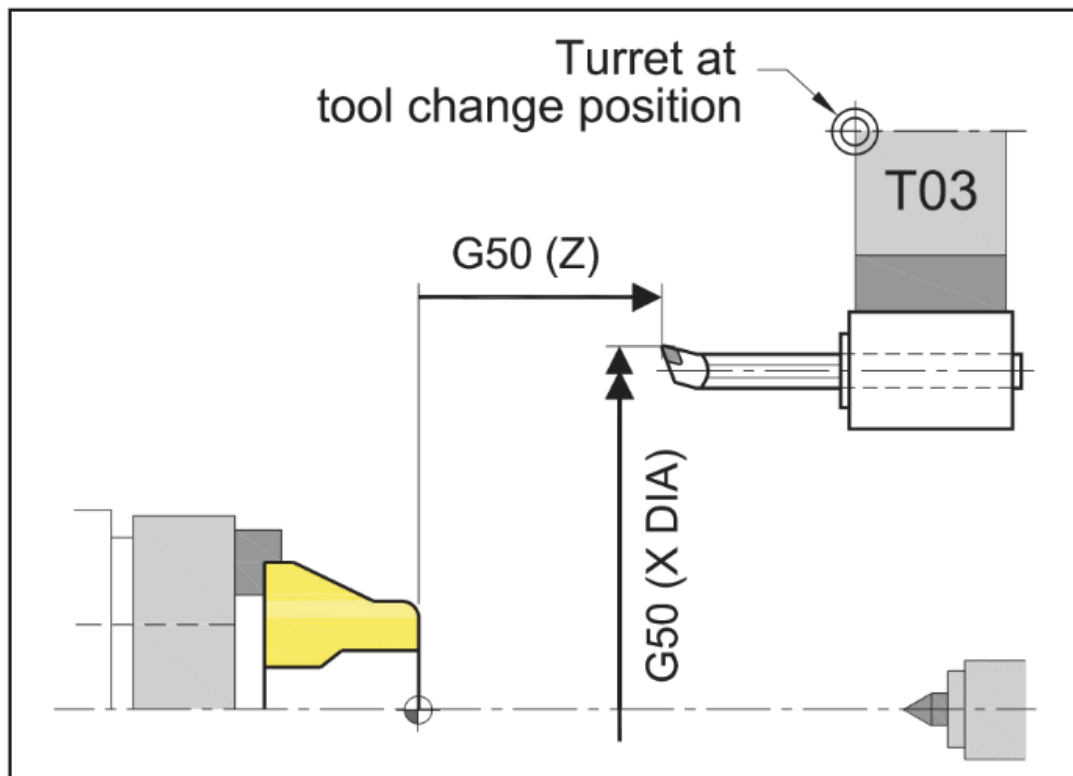


Figure 16-6

Typical G50 setting for internal lathe tools

For safety reasons, no tool should extend from a turret into Z negative zone - that is to the left of part front face. Many CNC lathes allow travel *beyond* the Z-axis machine zero (about 1-2 inches or 25-50 mm). Sometimes, this zone can be entered to make a safe tool change for very long tools. However, this is more advanced programming method and requires strict safety considerations. There is virtually no extended zone for X-axis above machine zero position (only about 0.02 inches or 0.5 mm).

Another safety concern relating to long tools is clearance in the part holding area, including chuck and jaws. Make sure to extend only those tools where the work requires it.

Corner Tip Detail

Typical turning tool contains an indexable insert, with a corner radius for strength and surface finish control. When position register command is used for a tool that has a radius built-in, the programmer has to know (and also tell the CNC operator), *which edge* the G50 setting corresponds to. In many cases, the choice is simple. G50 value is measured from program zero to the imaginary intersection of tangential X and Z-axis. Depending on actual tool shape and its orientation in the turret, G50 setting will vary. *Figure 16-7* on the next page shows several typical settings for the most common orientations of a tool with a corner radius, including two grooving tools.

Programming Example

The example showing how to use position register command G50 on a lathe will be very similar to that of a machining center. First, the tool change is made, followed with G50 setting for the selected tool. When machining is done with that tool, tool has to return to the same absolute position as specified in the G50 block. The following simplified example is for two tools - the first tool is programmed to cut a face; the second tool is programmed to cut a 2.5-inch diameter:

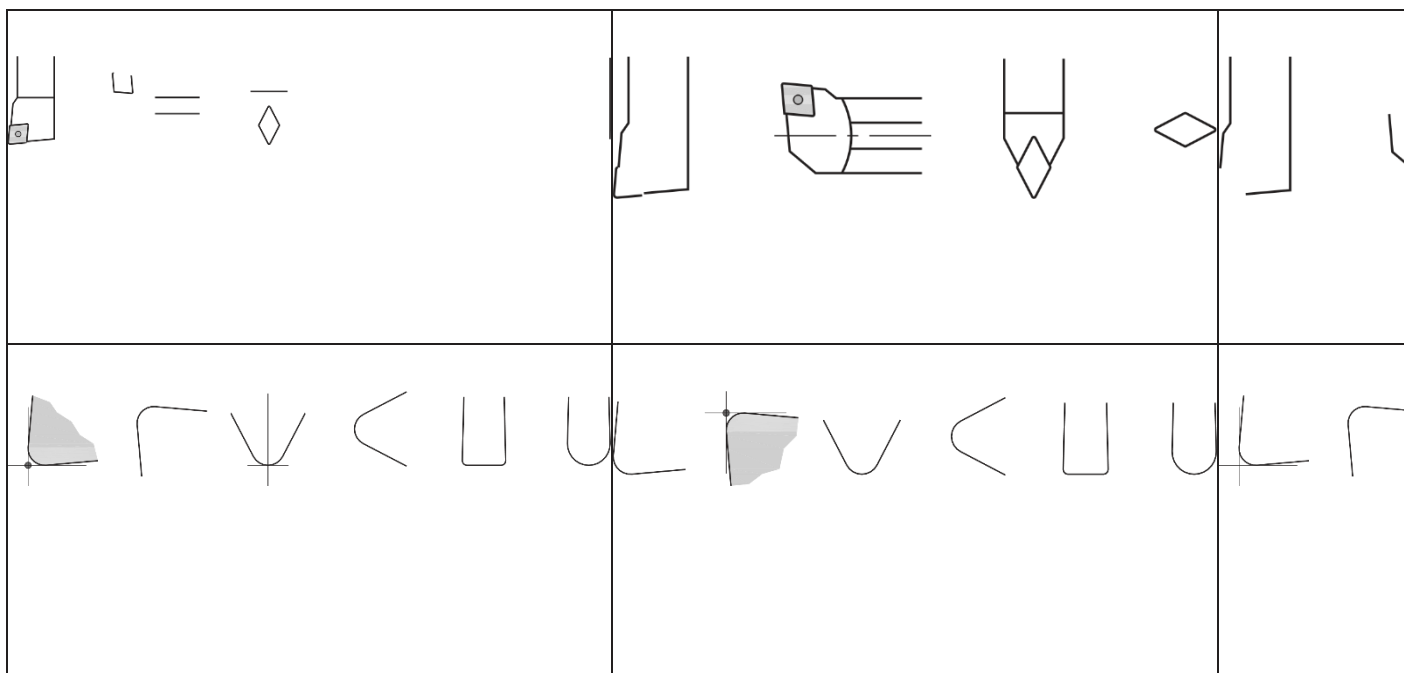


Figure 16-7

Position register setting G50 for common tool tip orientations - the heavy dot indicates XZ coordinates set by

O1602

N1 T0100

N2 G50 X7.45 Z5.5

N3 G96 S400 M03

N4 G00 X2.7 Z0 T0101 M08

N5 G01 X-0.07 F0.007

N6 G00 Z0.1 M09

N7 X7.45 Z5.5 T0100 N8 M01

N9 T0200

N10 G50 X8.3 Z4.8

N11 G96 S425 M03

N12 G00 X2.5 Z0.1 T0202 M08

N13 G01 Z-1.75 F0.008

N14 G00 X2.7 M09

N15 X8.3 Z4.8 T0200

N16 M30 %

Note blocks N2 and N7 for the first tool, and N10 and N15 for the second tool. For each tool, these XZ pairs of blocks are exactly the same. What the program is 'telling' the control system here is that block N2 only registers the current tool position, but block N7 actually returns that tool to the same position it came from. For the second tool, block N10 registers the current tool position, block N15 forces the tool to return there.

Other important blocks to consider together are blocks N7 and N10. Block N7 is the *tool change position* for the first tool, block N10 is the *tool position register* for the second tool - *both tools are at the same physical position of the turret!* The difference in XZ values reflects the difference in the extension (length) of each tool from the turret station. All that is done with G50 command is telling the control *where the current tool tip is from program zero* - always keep that in mind!

16. POSITION COMPENSATION

Obsolete subject - chapter included for compatibility purposes

Relationships between various reference points in CNC programming are always expressed as preset numeric values. Often, these numbers, these specific values, are required well *before* actual machine setup takes place. During programming, many dimensions are known exactly, others are known approximately and there are also those that are not known at all. Some known dimensions are subject to variations between jobs. Without any corrective facility available to the programmer, it will be almost impossible to setup the machine precisely and efficiently. Fortunately, modern controls offer many features to make both programming and machine setup an easier, faster and more precise activity. A number of coordinate systems, offsets and compensations are typical support tools used in programming for corrective purposes.

One of the oldest programming methods available is called *position compensation*. As the name suggests, using position compensation functions, the *actual* tool position is compensated relative to its *theoretical* or *assumed* position.

It is only one of several corrective methods available to the programmer. On modern CNC systems, this method is available for compatibility with older programs. Today, this technique is obsolete. It has been replaced by the much more flexible *Work Offsets (Work Coordinate System)*, described in the next chapter. This chapter describes some typical programming applications that can benefit from using the old-fashioned position compensation method.

A. DESCRIPTION

The main purpose of position compensation is to correct any differences between machine zero and program zero tool positions. In practice, it is used in those cases, where the distance between two reference points is subject to some variations or is not known at all. For example, when working with castings, program zero taken from the cast surface will be subject to frequent changes. Using position compensation will eliminate the need to make constant program changes or realignment of the fixture setup. Normally, a part is mounted in a fixture on the table and its whole setup is compensated. For this reason, position compensation is sometimes called *fixture offset* or the *table offset*. The difference between an offset and a compensation is often very subtle, and for any practical purposes, the two terms are same.

In this digital book, each term is used in the same meaning as the majority of users interpret it. Position compensation can also be used for a very limited replacement of the cutter radius offset - this usage is not covered at all for its obsolescence. Instead, the emphasis will be on positioning of tool from machine zero towards the part.

Like several other functions, position compensation is a programming method that requires input of the CNC machine operator. Programmer specifies the type of compensation and the memory register number, CNC operator enters actual setting values at the machine, using appropriate display screens, during part setup.

Programming Commands

On Fanuc and similar controls, there are four old preparatory commands (G-codes) available to program position compensation functions:

G45	<i>Single increase</i> in the programmed direction by the compensation amount
G46	<i>Single decrease</i> in the programmed direction by the compensation amount
G47	<i>Double increase</i> in the programmed direction by double the compensation amount

G48	<i>Double decrease in the programmed direction by double the compensation amount</i>

These definitions are based on *positive* compensation values stored in the control register. If the stored values are negative, the meaning of all definitions is valid only when the signs are inverted. None of these four preparatory commands is modal and all are valid only within the block in which they appear. If required in many blocks, they must be repeated in any subsequent block, if needed again.

Programming Format

Each G-code (G45 to G48) is associated with a unique position compensation number, programmed with the address *H*. The *H* address points to the control system memory area storage number. On most Fanuc controls, the programmed letter can also be *D*, with exactly the same meaning. Whether the *H* or *D* address is used in the program, depends on the actual setting of a control system parameter.

123

Atypical programming format for position compensation function is:

G91 G00 G45 X.. H..

or

G91 G00 G45 X.. D..

where the appropriate G-code (G45 through G48), is followed by the target position and number of the memory storage area (using *H* or *D* address).

Note that the example uses *incremental* and *rapid motion* modes and only *one axis*. Normally, the compensation has to be applied to both X and Y axes. However, only a single measured amount can be stored under either H or D number. Since it is most probable that the compensation value will be different for each axis, it must be specified on separate blocks, with two *different* offset numbers H (or offset numbers D), for example:

G91 G00 G45 X.. H31 (H31 STORES THE X VALUE)

G45 Y.. H32 (H32 STORES THE Y VALUE)

or

G91 G00 G45 X.. D31 (D31 STORES THE X VALUE)

G45 Y.. D32 (D32 STORES THE Y VALUE)

For the record, the H address is also used with another type of compensation, known as the *tool length offset* (or *tool length compensation*), described in *Chapter 19*. The D address is also used with another type of compensation, known as the *cutter radius offset* (or *cutter radius compensation*), described in *Chapter 30*.

The applicable preparatory G code will determine how the address H or address D will be interpreted. In the examples, more common address H will be used - *Figure 17-1*.

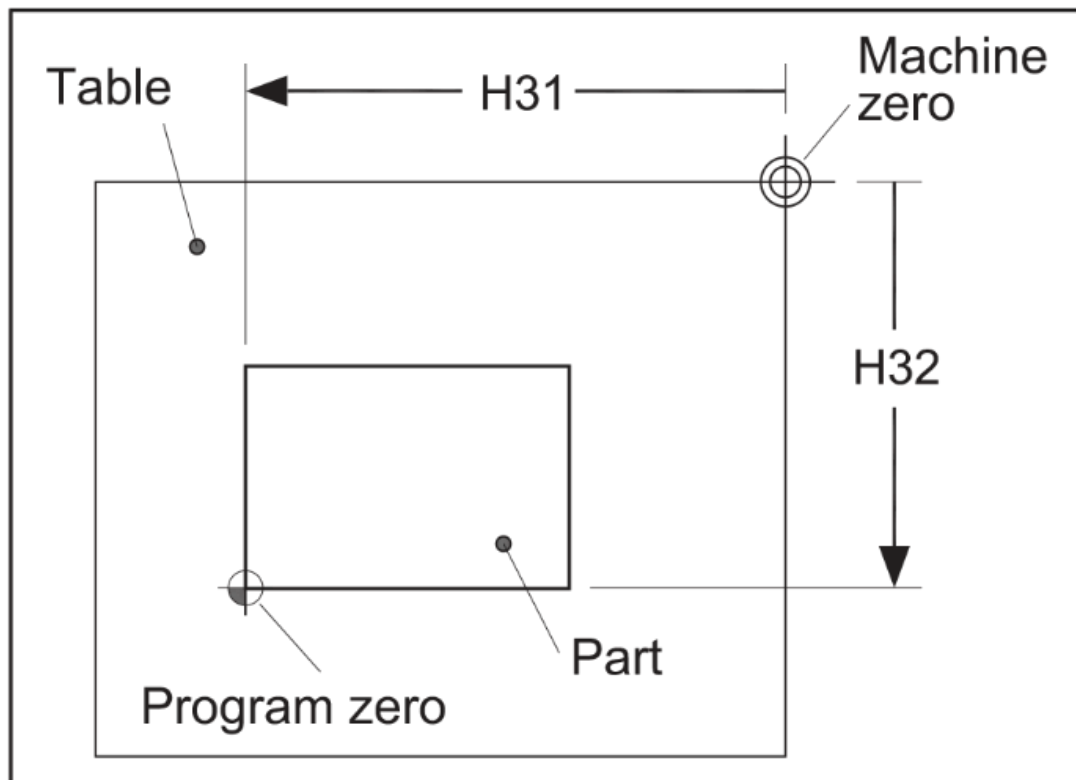


Figure 17-1

Position compensation - general concept

Chapter 17

Incremental Mode

A question may arise why the compensated motion is in incremental mode. Remember that the main purpose of position compensation is to allow a correction of the distance between machine zero and program zero. Normal use is when starting the tool motion *from* machine zero position. By default - and without any offsets - coordinate settings or active compensations, machine zero is the absolute zero, it is the *only* zero the machine control system ‘understands’ at the time.

Take the following example of several blocks, typically programmed at the beginning of a program with position compensation:

N1 G20

N2 G17 G80 T01

N3 M06

N4 G90 G00 G45 X0 H31 (NO X MOTION) N5 G45 Y0 H32 (NO Y MOTION)

N6 ...

This example illustrates a motion from machine zero (the current tool position), to program zero, which is the target position, along XY axes. Note the absolute mode setting G90 in block N4. Assume that the control system is set to H31 = -12.0000 inches. Control will evaluate the block and interpret it as programmer’s intention to go to the absolute zero, specified by G90. It checks the current position, finds that it is at the absolute zero already and *does nothing*. There will be *no motion*, regardless of the compensation value setting, if the absolute motion is programmed to either X0 or Y0 target position. If G90 is changed to G91, from absolute to incremental mode, there *will* be a motion along the *negative* direction of X-axis, by the distance of exactly 12 inches and there will be a similar motion along Y-axis, in block N5. The conclusion? *Use position compensation commands in the incremental mode G91 only.*

Motion Length Calculation

Look a little closer at how the control system interprets a position compensation block. Interpreting the way how the control unit manipulates numbers is important for understanding how a particular offset or compensation actually works. Earlier definition has stated that single increase is programmed with G45 command and single decrease with G46 command. Both G47 and G48 commands are of no consequence at the moment. Since both commands are tied up with a particular axis and with a unique H address, all possible combinations available must be evaluated:

Either an *increase* or a *decrease* is programmed

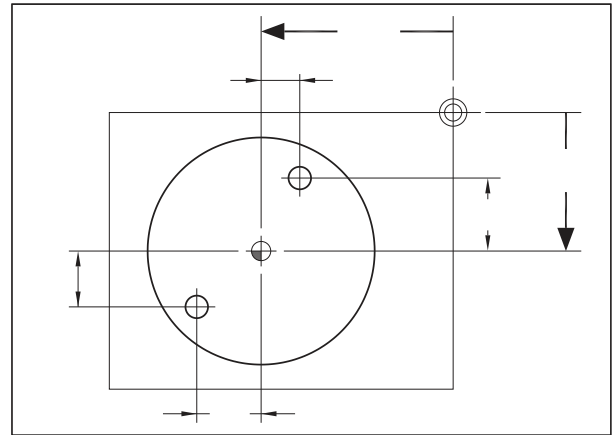
(G45 or G46)

Axis target can have a *zero* value, or a *positive* value, or a *negative* value

Compensation amount may have a *zero* value, or a *positive* value, or a *negative* value

POSITION COMPENSATION

In programming, it is important to set certain standards and consistently abide by them. For example, on vertical machining centers, compensation is measured *from* machine zero *to* program zero. That means a negative direction from the operator's viewpoint. The result is a logical decision to set negative compensation values as standard.



It is very crucial to understand *how* the control interprets information in a block. In position compensation, it evaluates the value stored in memory called by the address H (or D). If the value is zero, no compensation takes place. If the value of H is stored as a negative value, it *adds* this value to the value of axis target position and the result is the motion length and direction. For example, assume the memory register H31 stores the value of -15.0 inches, and the machine current location is at its zero position and the control axis setting is also set to zero. Then, the block

G91 G00 G45 X0 H31

will be interpreted as

-15.0 + 0 = -15.0000

resulting in the total motion of negative 15.0 inches along the X-axis.

If the setting of X-axis target position is a non-zero and positive, the same formula applies:

G91 G00 G45 X1.5 H31

will be interpreted as

-15.0 + 1.5 = -13.5000

However, the next example is *not* correct:

G91 G00 G45 X-1.5 H31

Here, the motion will try to go into the *positive* X-axis direction and the result will be X-axis overtravel. Since the value of X is negative, G45 command cannot be used and G46 command must be *used* instead:

G91 G00 G46 X-1.5 H31

will be interpreted as

$$\mathbf{-15.0 + (-1.5) = -15.0000 - 1.5 = -16.5000}$$

G45 could have been left in the program and the negative offset value could have been changed to a positive value. This could be quite confusing and definitely not consistent, but it would work quite well. To see the different possibilities, program O1701 is not doing very much, except moving from machine zero to different positions and back to machine zero (G28 command refers to a machine zero return and is explained separately, starting on *page 153*).

125

H98

9

H99

17

13

15

Figure 17-2

Position compensation applied to different target locations: zero, positive and negative - see O1701 program example

Figure 17-2 shows illustration for the following program example O1701. The logic applies to both X and Y axes exactly the same way. It is written in metric units and has been tested on Fanuc 11M, with the H address (D would work the same way). Position compensation values H98 and H99 were set to:

H98 = -250.000

H99 = -150.000

... for the X and Y axes respectively. Modal commands were not repeated:

O1701	(G45 AND G46 TEST)
N1 G21 G17	
N2 G92 X0 Y0 Z0	
N3 G90 G00 G45 X0 H98	(ABS X0 TARGET)
N4 G46 Y0 H99 N5 G28 X0 Y0	(ABS Y0 TARGET)
N6 G91 G00 G45 X0 H98	(INC X0 TARGET)
N7 G46 Y0 H99 N8 G28 X0 Y0	(INC Y0 TARGET)
N9 G90 G00 G45 X9.0 H98	(ABS X+ TARGET)
N10 G46 Y17.0 H99 N11 G28 X0 Y0	(ABS Y+ TARGET)
N12 G91 G00 G45 X9.0 H98	(INC X+ TARGET)
N13 G46 Y17.0 H99 N14 G28 X0 Y0	(INC Y+ TARGET)
N15 G90 G00 G45 X-15.0 H98	(ABS X- TARGET)
N16 G46 Y-13.0 H99 N17 G28 X0 Y0	(ABS Y- TARGET)

N18 G91 G00 G45 X-15.0 H98	(INC X- TARGET)
N19 G46 Y-13.0 H99	(INC Y- TARGET)
N20 G28 X0 Y0	
N21 M30	
%	

Control system will process each motion block individually - either the way it was intended or the wrong way (symbol O/T means an *overtravel* condition, preceded with the axis and direction of the overtravel):

N3 G90 —> G45 —> 0 ... no motion
N4 G90 —> G46 —> 0 ... no motion
N6 G91 —> G45 —> 0 ... X-250.0
N7 G91 —> G46 —> 0 ... Y+ O/T
N9 G90 —> G45 —> + ... X-241.0
N10 G90 —> G46 —> + ... Y+ O/T
N12 G91 —> G45 —> + ... X-241.0
N13 G91 —> G46 —> + ... Y+ O/T
N15 G90 —> G45 —> - ... X+ O/T
N16 G90 —> G46 —> - ... Y-163.0
N18 G91 —> G45 —> - ... X+ O/T N19 G91 —> G46 —> - ... Y-163.0

Position Compensation Along the Z-axis

Position compensation feature usually applies only to X and Y axes and will not normally be used with the Z-axis. In most cases, the Z-axis has to be controlled by another kind of compensation - known as the *tool length offset*.

This very current and modern method is described in detail, starting on *page 135*. If the Z-axis is programmed with G45 or G46 commands, it will also be affected.

Using G47 and G48

In the examples, position compensation feature was used only between the machine zero and program zero, as a method of determining where exactly is the part located on the table. Single increase using G45 and single decrease using G46 commands were used, because they were the only commands needed.

Commands G47 (double increase) and G48 (double decrease) are only necessary for a very simplified cutter radius offset and are not covered in this digital book because of their obsolescence. However, they can still be used.

Face Milling - One Possible Application

In a later section (*page 235*), the principles of face milling will be explained in more detail. In that chapter is a very good example of how to apply position compensation to offset diameter of the face mill in a clear position, regardless of its size. This is probably the only possible use of G45 and G46 commands in contemporary programming.

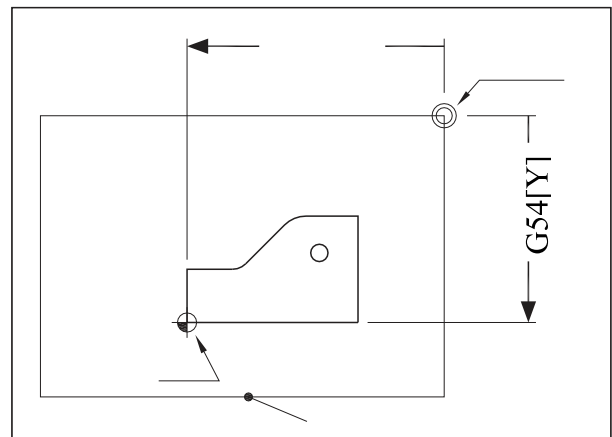
17. WORK OFFSETS

Using the method of *Work Offsets* for tool positioning based on machine zero is much faster and more efficient than using the older methods of position compensation functions G45 and G46 described in the previous chapter. Work offsets are also known as *Work Coordinate System*, or even as *Fixture Offsets*. Work offsets are much more efficient than using the position register commands G92 (milling systems) or G50 (turning systems). CNC programmers who do not know the meaning of *position compensation* functions or the meaning of *position register* commands, are most likely working with the most modern CNC machines only. However, there are many machines in industry that still require these rather obsolete functions. Knowing them well will increase the number of available programming tools.

This chapter describes the most modern methods to coordinate the relationship between machine zero reference position and the program zero reference point. Focus will be on the *Work Coordinate System* feature of a modern control, whether it is called the *Work Coordinate System* or the *Work Offsets*. The latter term seems to be more popular because it is a little shorter. Think of the work offsets as an *alignment* between two or more coordinate systems.

A. WORK AREAS AVAILABLE

Before some more detailed descriptions can be covered, just what *is* a work coordinate system - or a *work offset*?



Work offset is a method that allows the CNC programmer to program a part away from CNC machine, without knowing its exact position on the machine table. This is a very similar approach as in the position compensation method, but much more advanced and flexible. In work offset system, up to six parts may be set up on the machine table, each having a different work offset number. Programmer can move the tool from one part to another with absolute ease. To achieve this goal, a special preparatory command for the active work offset is needed in the program and the control system will do the rest. Control system will automatically make any adjustment for the difference between the two-part locations.

Unlike position compensation function, two, three, or more axes may be moved simultaneously with work offsets, although the Z-axis for CNC machining centers is controlled independently, using G43 or G44 tool length offset commands. Commands relating to the Z-axis offset are fully described in the next chapter - see page 135.

In position compensation, to switch machining from one part to another within the same setup, the program has to contain a different compensation number from program zero of the previous part. Using the work offset method, *all program zeros are measured from machine zero position*, normally up to six, but more offsets are available.

The six work coordinate systems - or *work offsets* - that are available on Fanuc control systems are assigned the following preparatory commands:

G54	G55	G56	G57	G58	G59

When the control unit is turned on, the default coordinate system is normally G54, at least in most cases.

Basically, work offsets establish up to six independent work areas as a standard feature. Settings stored in the CNC unit are always distances measured *from machine zero to program zero*. As there are up to six work areas, up to six independent program zero positions can be defined. *Figure 18-1* shows the basic relationships, using the default G54 setting.

G54 [X] Machine

zero

PART

Program zero

Axes motion limits

Figure 18-1

Basic relationships of the work offset method

The same relationships illustrated for default work offset apply exactly the same way for the other five available work offsets G55 to G59. Settings stored in the control system are always physically measured from machine zero position *to* program zero of the part, as determined by CNC programmer. Operator uses edge finders, corner finders and other devices to find these distances.

The distance *from* machine zero *to* program zero of each work area is measured separately along the X and Y axes and input into the appropriate work offset register of the control unit. Note that the measurement direction is *from* machine zero *to* program zero, never the other way around. If the direction is negative, the minus sign must be entered in the offset screen.

For comparison with the position register command G92, *Figure 18-2* shows the same part set with the older method of G92 and machine zero as a start point. Note the *opposite* arrows designation, indicating the direction of measurement - *from* program zero *to* machine zero.

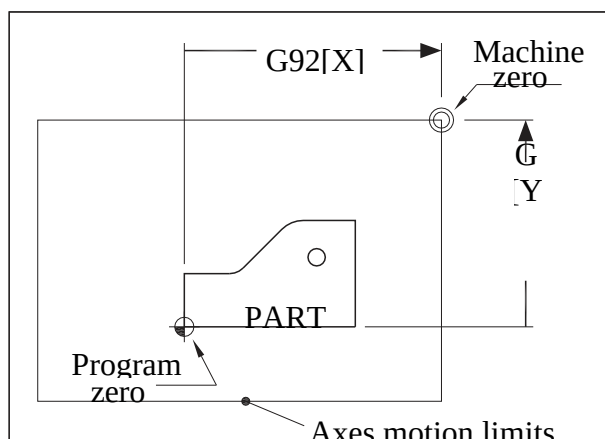


Figure 18-2

Basic relationships of the Position Register command G92

For work offsets G54 to G59, a typical entry into the coordinate offset position register will be the X-axis as a negative value, the Y-axis as a negative value and the Z-axis as a zero value, for the majority of vertical machining centers. This is done by CNC operator at the machine. *Figure 18-3* shows an example of a typical control system entry.

01 (G54)
X -12.5543 Y -
.4462
Z
0.0000

Figure 18-3

Typical data entry for the G54 work coordinate system

By using G54 to G59 settings in the program, control system selects the stored measured distances, and the cutting tool may be moved to any position within the selected work offset simultaneously in both X and Y axes, whenever desired.

Part position on the machine table is usually unknown during programming process. The main purpose of work offset is to synchronize the actual position of the part as it relates to the machine zero position.

Additional Work Offsets

The standard number of six work coordinate offsets is usually enough for most types of work. However, there are jobs that may require machining with more program reference points, for example, a multi-sided part on a horizontal machining table. What options do exist, if the job requires ten work coordinate systems, for example?

Fanuc offers - *as an option* - up to 48 additional work offsets, for the total of 54 (6+48). If this option is available on the CNC system, any one of the 48 work offsets can be accessed by programming a special G code:

G54.1 P..	Selection of additional work offset, where P = 1 to 48

G54.1 P.. example:

G54.1 P1	Selection of additional work offset 1
G54.1 P2	Selection of additional work offset 2
G54.1 P3	Selection of additional work offset 3
G54 1 Px..	Selection of additional work offset x..
G54.1 P48	Selection of additional work offset 48

The utilization of additional work offsets in the program is exactly the same as that of standard commands:

N2 G90 G00 G54.1 P1 X5.5 Y3.1 S1000 M03

Most Fanuc controls will allow omission of the decimal portion of the G54.1 command. There should be no problem programming:

N2 G90 G00 G54 P1 X5.5 Y3.1 S1000 M03

The presence of P1 to P48 function within a block will select an *additional* work offset. If P1 to P48 parameter is missing, the default work offset command G54 will be selected by the control system.

B. LATHE APPLICATIONS

Originally, work coordinate system was designed for CNC machining centers only. It did not take long to apply it to CNC lathes as well. The operation, logically and physically, is identical to

that for machining centers. Using work offsets for CNC lathes eliminates the awkward use of G50 or G92 and makes CNC lathe setup and operation much faster and easier.

Types of Offsets

The main difference in applying work offsets on a lathe is that seldom will there be a need for more than one work offset. Two work offsets are a possibility, three or more are used for some very special and complex setups. G54 to G59 commands are available on all modern CNC lathes and it is quite customary to ignore work offset selection in the program, unless more than one offset is used. That means the CNC lathe programmer depends on the default G54 setting as a rule.

Two special offset features found on the latest control systems are *Geometry* and *Wear* offsets, either on the same screen display, or on separate screens, depending on the control model.

Geometry Offset

Geometry offset is the equivalent of a work offset as it is known from milling controls. It represents the distance from tool reference point to program zero, measured from machine zero along a selected axis. Typically, on a slant bed CNC lathe, with tool turret *above* the spindle centerline, the geometry offset for both X and Z axes will be negative. *Figure 18-9* illustrates reasonable geometry values for a drill, turning tool and boring bar (T01, T02, T03).

GEOMETRY OFFSET				
No.	X-offset	Z-offset	Radius	Tip
01	-8.6330	-2.3630	0.0000	0
02	-8.6470	-6.6780	0.0469	3
03	-9.0720	-2.4950	0.0313	2
04	0.0000	0.0000	0.0000	0
05	0.0000	0.0000	0.0000	0

Figure 18-9

Typical data entries for a lathe tool GEOMETRY offset

Wear Offset

Wear offset is also known and used on milling controls, but only for the tool length offset and cutter radius offset, not for work coordinate system (work offset).

On CNC lathes, the purpose of wear offset is identical to that for machining centers. This offset compensates for the tool wear and is also used to make fine adjustments to the geometry offsets. As a rule, once the geometry offset for a given tool is set, that setting should be left unchanged. Any adjustments and fine tuning of actual part dimensions should be done by the wear offset only.

WEAR OFFSET				
No.	X-offset	Z-offset	Radius	Tip
01	0.0000	0.0000 0.0000	0.0000	0
02	-0.0060	0.0040	0.0469	3
03	0.0000	0.0000	0.0313	2
04	0.0000	0.0000	0.0000	0
05	0.0000		0.0000	0

Figure 18-10

Typical data entries for a lathe tool WEAR offset

Figure 18-10 shows some reasonable sample entries in the wear offset registers. The tool radius and tip number settings appear in both displays and the display in both screens is automatic after the offset value input. Tool nose radius and tool tip orientation number are unique to CNC lathe controls.

Tool and Offset Numbers

Just like tools on CNC machining centers have numbers, they have numbers on CNC lathes as well. Usually, only one coordinate offset is used, but different tool numbers. Remember, the tool number for a lathe has *four digits*, for example, T0404:

The first two digits select the tool indexing station (turret station) and the geometry offset number. There is no choice here. Tool in station 4, for example, will also use geometry offset number 4

The second two digits are for the wear offset register number only. They do not have to be the same as the tool

number, but it makes sense to match the numbers, if possible

Depending on the control model and the display screen size, tool offset register may have a separate screen display (page) for geometry and wear offsets, or both offset types may be shown on the same screen display. Work offset settings (work coordinates) are always placed in the *geometry* offset column.

C. TOOL SETUP

In the next three illustrations is a very similar layout as that shown on *page 119*, describing the use of G50 register method (position register command used in the program). *Compare the two illustrations!*

Setup of a CNC lathe is identical in both cases, except for the method and purpose of tool position measuring. All illustrations in the applications also match the reasonable data entered in tool geometry and tool wear offset screens of the control.

Typical settings along the X-axis are always negative (as shown in illustrations), typical settings along the Z-axis are usually negative. A positive value is also possible, but that means the tool is above work and tool changing can be very dangerous. *Watch out for such situations!*

The actual setting procedures are subject of a CNC machine operation training and not practical to cover in a programming digital book. There are additional methods, also part of machine training, that allow faster tool setting, using one tool as a primary and setting all the remaining tools *relative to the primary tool*.

Center Line Tools

Tools that work on the spindle center line are tools that have their tool tip located on the center line during machining. This area covers all center drills, spot drills, various drills, reamers, taps, even end mills used for flat bottom holes. At the same time, it disqualifies all boring bars, since their tool tip does not normally lie on the spindle center line during machining. Center line tools are *always* measured from the center line of the tool to the center line of the spindle along the X-axis and from the tool tip to program zero along the Z-axis. *Figure 18-11* illustrates a typical setting for center line tools.

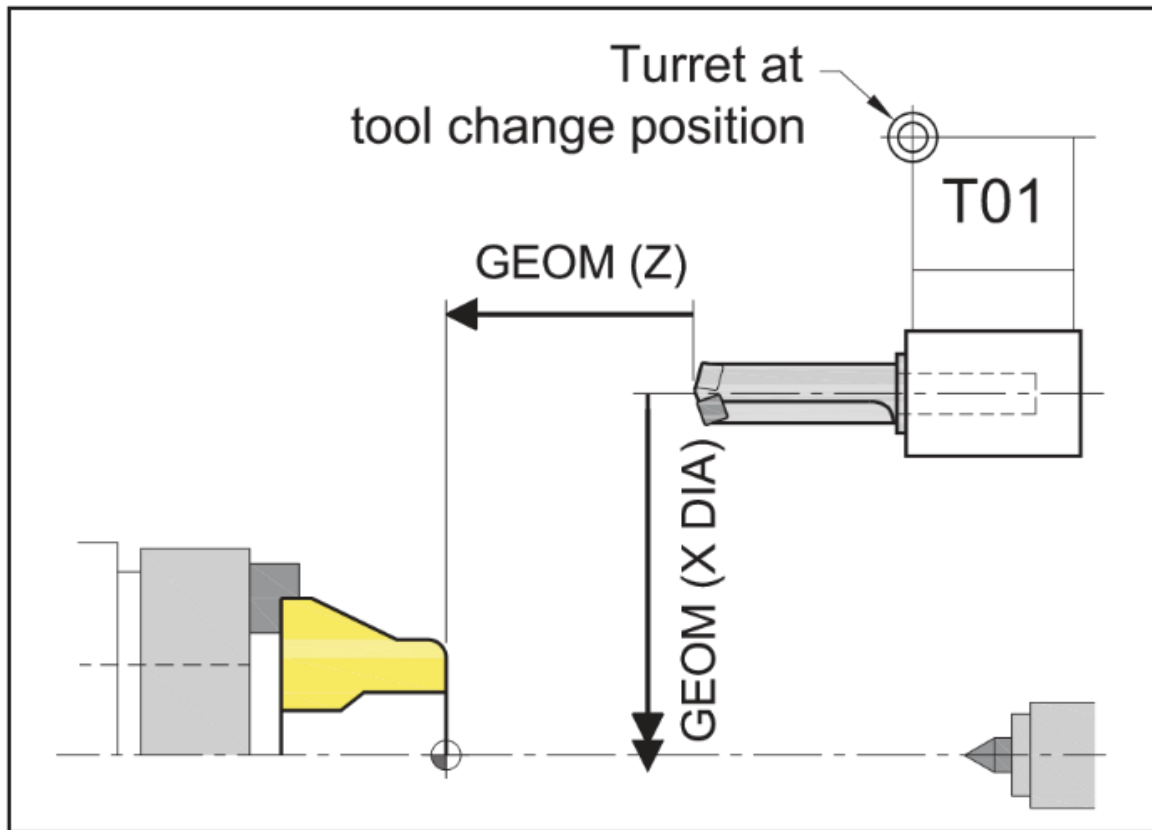


Figure 18-11

Typical geometry offset setting for CENTER LINE tools

Turning Tools

Turning tools - or *external* tools - are measured from the imaginary tool tip to program zero, along the X-axis (as a negative diameter) and along the Z-axis, usually as a negative value as well. Keep in mind that if the cutting tool insert (for turning or boring) is changed from one radius to another radius in the *same* tool holder, the entered setup amount must also change. Such a change may be marginal, but even a marginal change is enough to cause a scrap, so a good care is needed. For turning, be extra careful about a tool nose radius that changes from a larger size to a smaller size, for example, from 3/64 (R0.0469) to 1/32 (R0.0313).

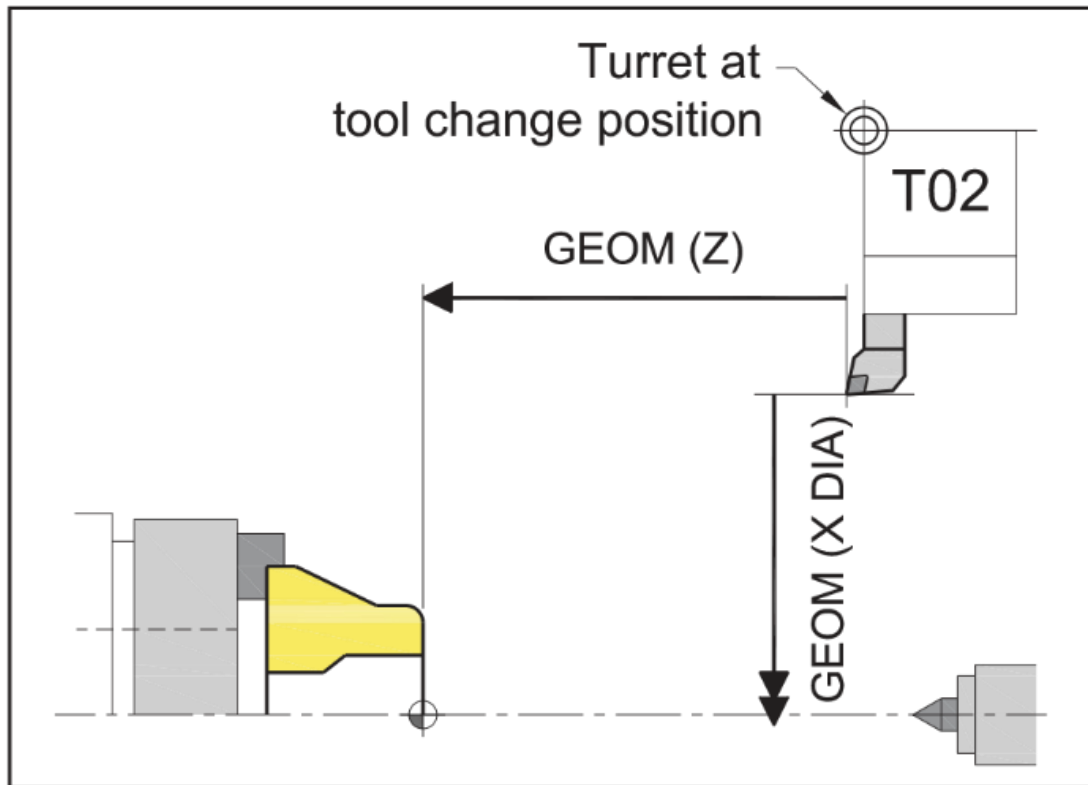


Figure 18-12

Typical geometry offset setting for EXTERNAL tools

Figure 18-12 illustrates a typical geometry setting for a turning (external) tool and Figure 18-13 illustrates a typical geometry setting for a boring (internal) tool.

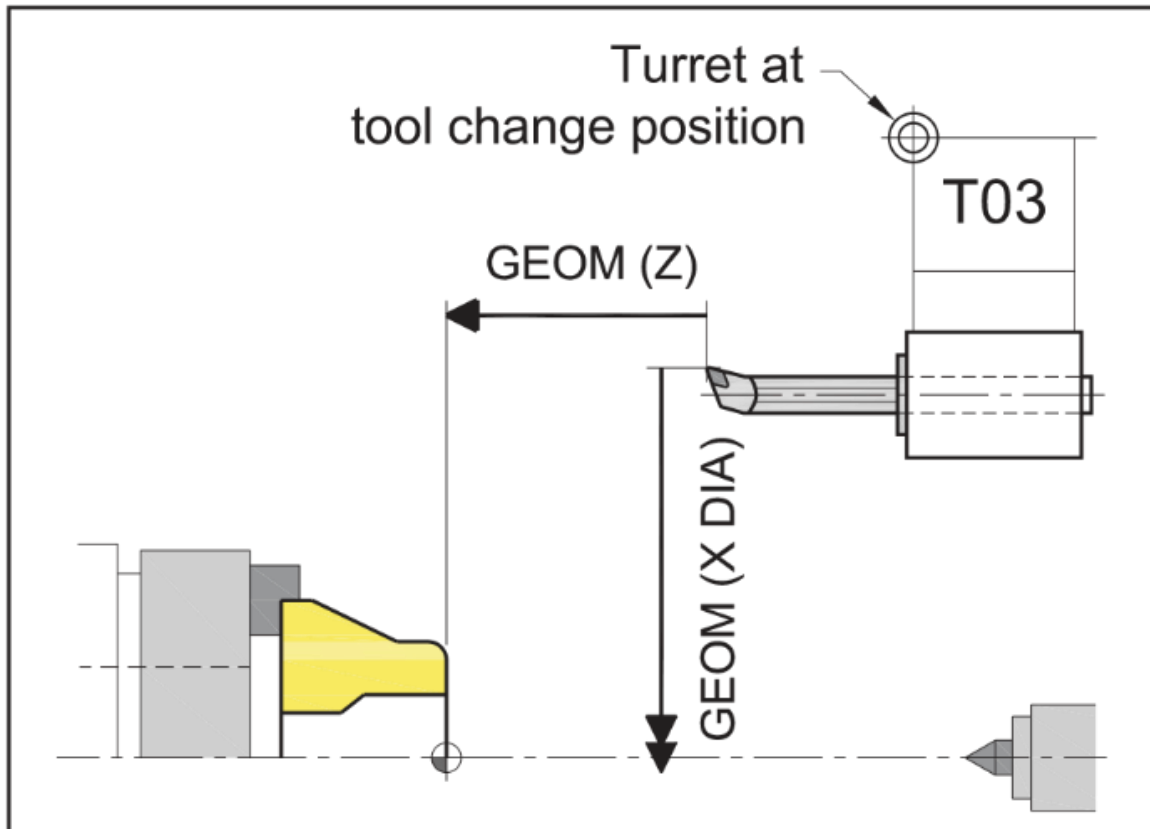


Figure 18-13

Typical geometry offset setting for INTERNAL tools

Boring Tools

Boring tools - or *internal* tools - are always measured from the imaginary tool tip to program zero, along X-axis (typically as a negative diameter) and along Z-axis, typically as a negative value as well. In majority of cases, the X setting amount of a boring tool will be noticeably larger than that for a turning or other external tool.

For boring operations, same as for turning operations, also be extra careful about a tool nose radius that changes from a larger size to a smaller size. It is the same as for a turning tool. Scrap can be made very easily.

Command Point and Tool Work Offset

For various reasons, it is quite common to change a cutting insert in the middle of work, primarily to maintain favorable cutting conditions and to keep dimensional tolerances within drawing specifications. Cutting inserts are manufactured to very high standards, but a certain tolerance deviation should be expected between inserts obtained from different sources. If changing an insert, it is advisable to adjust the wear offset for precision work, in order to prevent scrapping the part.

Tool holders accept inserts of the same shape and size but with a different nose radius. Always be cautious when replacing an insert with an insert that has a larger or smaller tool nose radius. The offset has to be adjusted in both axes, by the proper amount.

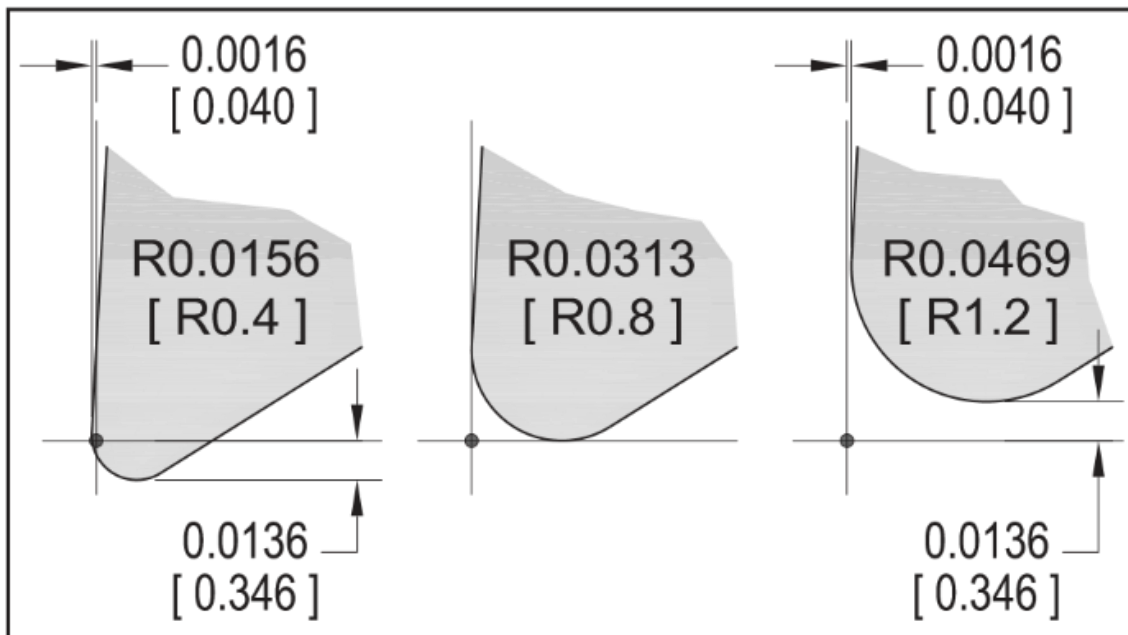


Figure 18-14

Setting error caused by a different insert radius in the same holder

Figure 18-14 example shows the most common standard setting for a 1/32 (0.0313) [0.8 mm] nose radius (middle), and the setting error for a radius that is smaller (left) and one that is larger (right). Dimensions indicate the amount of error for the particular insert shown in the example. Metric dimensions are shown in the brackets.

When changing an insert, adjust the required offset(-s)

18. RAPID POSITIONING

A CNC machine tool does not always cut material and 'make' chips. From the moment the cutting tool becomes active in a part program, it goes through a number of motions - some are productive (cutting), others are nonproductive (positioning).

Positioning motions are necessary but nonproductive. Unfortunately, these motions cannot be totally eliminated and have to be managed as efficiently as possible. For this purpose, the CNC system provides a feature called the *rapid traverse* motion. Its main objective is to shorten the positioning time between non-cutting operations, where the cutting tool is not in contact with the part. Rapid motion operations usually involve four types of motion:

From the tool change position towards the part

From the part towards the tool change position

Motions to bypass obstacles

Motions between different positions on the part

A. RAPID TRAVERSE MOTION

Rapid traverse motion, sometimes called a *positioning motion*, is a method of moving the cutting tool from one position to another position at a *rapid rate of the machine*. The maximum rapid rate is determined by CNC machine manufacturer and takes place within the machine travel limits.

The common rapid rate for many larger CNC machines is about 25000 mm/min (985 in/min). Medium and smaller machines can go much higher, offering rapid motion up to 75000 mm/min (2950 in/min) or even more, particularly for smaller machines. Machine manufacturer determines the rate of rapid motion for each of the machine axes. The motion rate can be the same for each axis or it can be different. A different rapid rate is usually assigned to the Z-axis, while X and Y axes have the same rapid motion rate.

Rapid motion can be executed as a single axis motion, or as a compound motion of two or more axes simultaneously. It can be programmed in absolute or incremental mode of dimensioning, and it can be used whether the spindle is rotating or stationary. During program execution, CNC operator may temporarily interrupt the rapid motion by pressing Feed hold key on the control panel, or even setting the feed rate override switch to zero or a decreased rate. Another kind of rapid rate control can be achieved by the *dry run* function, usually during setup.

G00 Command

Preparatory command G00 is required in CNC program to initiate rapid motion mode. Feed rate function F is *not required* with G00 and, if programmed, will be ignored during rapid motion (in G00 mode). Such a feed rate will be stored in memory and becomes effective beginning with the first occurrence of any cutting motion (G01, G02, G03, etc.), unless a new F-function is programmed together with the cutting motion:

Example A (feed rate in in/min):

N21 G00 X24.5 F30.0

N22 Y12.0

N23 G01 X30.0

In block N21, only the rapid motion will be executed. Feed rate of 30.0 in/min will be ignored in this block but stored for later use. Motion in the block N22 will also be in rapid positioning mode, since G00 is a modal command. The last block, N23, is a linear motion (cutting motion), that requires a feed rate. As there is no feed rate assigned to the motion in this block, the *last programmed* feed rate will be used. That was specified in block N21, and it will become the *current*

feed rate in block N23, as F30.0.

Example B (feed rate in in/min):

N21 G00 X24.5 F30.0

N22 Y12.0

N23 G01 X30.0 F20.0

In block N21, command G00 becomes modal, which means that it remains in effect until it is canceled by another command of the same group. In the example (B), command G01 in block N23 cancels rapid motion and changes rapid mode to a linear mode. Also, the feed rate is reprogrammed and will be at 20.0 in/min starting at block N23. In fact, the feed rate F30.0 in block N21 has never been used in the program - it is harmless but redundant and should be removed.

Rapid traverse motion is measured as the *distance in current units traveled in one minute* (measured in *mm/min* or *in/min*). The maximum rate is always set by the machine manufacturer, *never* by the control system or the program. A typical limit set by the machine builder is a rate between 7500 and 75000 mm/min (~300-3000 in/min), and even higher. Since motion per time is independent of the spindle rotation, it can be applied at any time, regardless of the last spindle rotation function mode (M03, M04, M05).

Depending on the CNC machine design, rapid motion rate can be the same for all axes, or each axis can have its own maximum rate. Maximum rapid rates for a typical machining center may be 30000 mm/min (1180 in/min) for the X and Y axes and about 24000 mm/min (945 in/min) for the Z-axis. For a CNC lathe, the rates are very similar, for example, 25000 mm/min (985 in/min) for the X-axis, and 30000 mm/min (1180 in/min) for the Z-axis. Emerging technologies allow for much higher rapid rates.

B. RAPID MOTION TOOLPATH

Every motion in G00 mode is a rapid, non-circular motion - circular and helical motions cannot normally be made at a rapid rate. The actual linear motion of a tool between two points is not necessarily the shortest toolpath in the form of a straight line, although many latest machines do provide such feature. On many CNC machines, a programmed toolpath and the resulting actual toolpath will be different, depending on several factors:

Number of axes programmed simultaneously

Actual length of motion for each axis

Rapid traverse rate of each axis

Since the only purpose of rapid motion is saving unproductive time (motion from the current tool position to the target tool position), toolpath itself is irrelevant to the shape of the machined part. Always be aware of the actual rapid toolpath motion for reasons of safety, particularly when two or more axes are programmed at the same time. No physical obstacles must be in the way of a toolpath motion.

If there is an obstacle between any two points of the toolpath, such obstacle *will not be* automatically bypassed by the control for one very simple reason - the control has no way of detecting any obstacle. It is the programmer's responsibility to assure that any tool motion (rapid motion included) occurs without any obstacles in its way.

Some typical examples of physical obstacles that can interfere with a tool motion are:

FOR MACHINING CENTERS:

Clamps, vises, fixtures, rotary or indexing table, machine table, part itself, etc.

FOR LATHES:

Tailstock quill and body, chuck, steady rest, live center, face plate, fixture, other tool, part itself, etc.

Additional obstacles during a tool motion may be caused by special types of setups, machine design, tool mounting method, etc.

Always watch for obstacles during rapid motion

Although an obstacle may be in the way of cutting motions in G01, G02 or G03 mode (for example a face turning towards a tailstock on a lathe), the most problems occur during rapid motions G00, G28, G29, G30 and with fixed cycles G81 to G89, G73, G74 and G76. During rapid motion, the actual toolpath on many machines is much less predictable than during cutting motions. Keep in mind that the only purpose of rapid motion is to get from one part location to another location *fast* - but not necessarily straight.

In order to bypass obstacles and still assure a safe rapid motion in the program at all times, let's take a closer look at the available options while programming a rapid motion.

Single Axis Motion

Any tool motion programmed specifically for only one axis at a time is always a straight line along the selected axis. In other words, each rapid motion that is parallel to one of the available axes, must be programmed in a separate block. The resulting motion is always equivalent to the shortest distance between the start and end points of this motion - *Figure 20-1*.

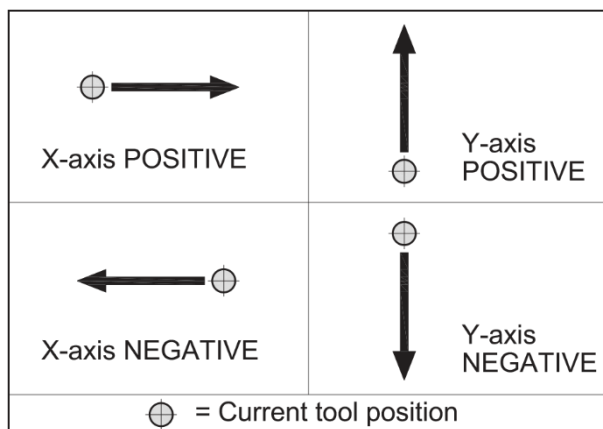


Figure 20-1

Single axis motion for a machining center application (XY shown)

Several consecutive program blocks, each containing only a single axis motion, can be included in the program to bypass obstacles to machining. This method of programming is preferable in cases where only the *exact* or *approximate* position of certain obstacles (such as clamps or fixtures) is known during part program preparation.

Multi-axis Motion

Cutting tool moved at rapid rate always uses the G00 command. If this motion is a motion of two or more axes simultaneously, the programmed rapid path between two points and the actual rapid path may not always be identical. The result is a compound motion that can be - and often is - much different from the theoretical programmed motion - the motion originally intended.

RAPID POSITIONING

In theory, the motion along any two axes is equivalent to a *straight diagonal motion*. The real motion, however, may or may not be a straight diagonal tool path at all. Consider the following example in *Figure 20-2*.

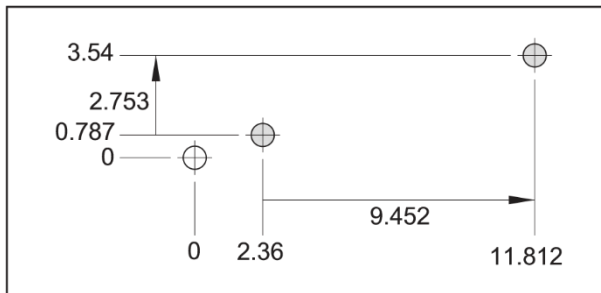


Figure 20-2

Drawing sketch for rapid motion examples

Current tool position (start point) is at X2.36 Y0.787 coordinate location. The tool motion terminates at X11.812 Y3.54 location. In terms of incremental motion, the cutting tool has to travel 9.452 inches along the X-axis and 2.753 inches along the Y-axis.

If rapid rate for both axes is the same (XY rapid motion rates usually are), such as 985 in/min, it will take

$$(9.452 \text{ 60}) / 985 = 0.576 \text{ seconds}$$

to complete the X-axis motion - but only

$$(2.753 \text{ 60}) / 985 = 0.168 \text{ seconds}$$

is required to complete the Y-axis motion. Since this motion is not completed until *both* axes reach the end point, it is logical that the actual tool path will be *different* from the programmed toolpath.

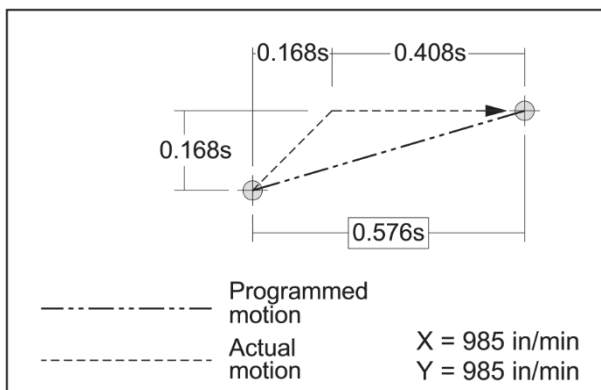


Figure 20-3

Rapid motion deviation - same rapid rate for each axis

Figure 20-3 shows a combination of an angular and a straight motion as the actual toolpath. Tool departs at the rate of 985 in/min (25000 mm/min) simultaneously in *both*

axes, with a resulting 45 motion. The total time required to reach the end position is 0.576 seconds, which is the longest time required for *either* axis to reach its target position. After the elapsed time of 0.168 seconds, the Y-axis target position has been reached, but there is still another 0.408 seconds left to complete the X-axis motion. The target position must be reached in *both* axes, so the tool then continues along X-axis only (for the remaining 0.408 seconds), in order to reach the final target position.

Another example, using the same location coordinates as in Figure 20-2, illustrates a less common situation, when the rapid rate is *different* for each axis - Figure 20-4.

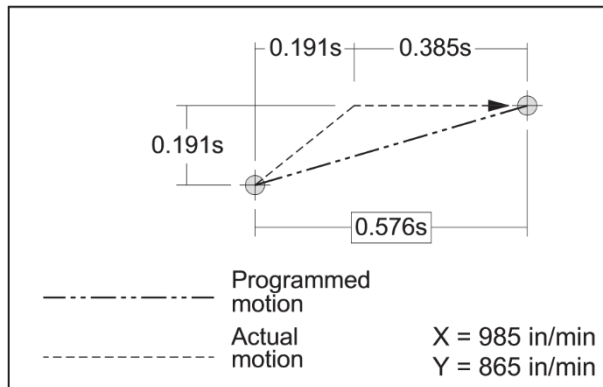


Figure 20-4

Rapid motion deviation - different rapid rate for each axis

In this not so common example, the X-axis rate is set to 985 in/min (25000 mm/min), but the Y-axis rate is set to 865 in/min (22000 mm/min). For the X-axis, it will then take

$$(9.452 \text{ 60}) / 985 = 0.576 \text{ seconds}$$

to complete the X-axis motion (no change) - but only

$$(2.753 \text{ 60}) / 865 = 0.191 \text{ seconds}$$

to complete the Y-axis motion. In this case, the resulting motion will also include an angular departure, but *not* at 45, because of the different rating of rapid traverse rate for each axis. During the 0.191 seconds (which is the common time to both axes), the X-axis motion will travel

$$0.191 / 60 \cdot 985 = 3.136 \text{ inches}$$

but the Y-axis motion will be only

$$0.191 / 60 \cdot 865 = 2.753 \text{ inches}$$

The resulting motion is at 41.279° with a slight rounding applied. Actual departure angle is not always necessary to be known, but it helps to calculate it for rapid motions in some very tight areas of a part. It only takes a few simple trigonometric functions to make sure of the true rapid toolpath, provided the rapid rate per axis is known.

Both of the above examples illustrate an angular motion along two axes, followed by a straight single axis motion in the remaining axis. The graphical expression of these motions is a bent line, resembling a *hockey stick* or a *dog leg* shape, which are also very common terms applied to such a rapid motion.

Calculation of the actual motion shape, as was done earlier, is only seldom necessary. Taking some basic precautions, rapid motion can be programmed safely *without* any calculations. If no obstacle is within the work area (an imaginary rectangle created by the diagonally positioned start and end point), there is no danger of collision due to the diverted rapid tool path. On CNC milling systems, the third axis can also be used. The rectangle of the above example will be enhanced by the third dimension and a three-dimensional space must be considered. In this case, no obstacle should be *within this space*, otherwise the same rules apply for a rapid motion along three axes as for a two-axis simultaneous rapid motion. Note that the rapid rate for the Z-axis on CNC machining centers is usually *lower* than the rapid rate for X and Y axes.

Straight Angular Motion in Rapid Mode

In some uncommon circumstances, the theoretical rapid toolpath will correspond to the actual toolpath (with no bent line as a result). This will happen if the simultaneous tool motion has the same length in each axis and the rapid rates of all axes are identical. Such an occurrence is rather rare, although not impossible. Some machine manufacturers provide this feature as a standard and the programmer should know whether the machining center does have that feature or not. Another

situation where the resulting motion is a straight angle, is when the rapid rating varies for each axis, but the required length of motion just 'falls' into the range that results in a straight angular motion.

Both of these occurrences are rare (more or less a case of good luck) and in actual programming will seldom happen. To be on the safe side, never take any chances - it is always more practical to program the rapid motion without the actual calculation of the toolpath but with safety as a primary consideration.

Reverse Rapid Motion

Any rapid motion must be considered in terms of *approach* towards a part and the *return* back to the tool changing position. This is the way a cutting tool is normally programmed - it starts at a certain position and then returns there, when all cutting activity for the tool is completed. It is not a mandatory method, but it is an organized method, it is consistent, and it makes programming much easier.

So far, the subject was a rapid motion *before* an actual cut, starting from the tool change position. When the tool cutting function is completed, a rapid motion is required to return back to the tool change position.

This consideration is more important in turning applications than in milling, due to the nature of programming for the two types of machines. In turning, the approach motion may be along the Z-axis first, to avoid a collision with the tailstock, and *then* along the X-axis. The reverse motion should be along the X-axis first, *then* along the Z-axis motion, in order to achieve the same safety goal when returning to the tool change position.

Atypical application of this programming technique may be useful after using a machining cycle (such as turning, boring, facing, threading, etc.), where the cycle starting point is also its end point.

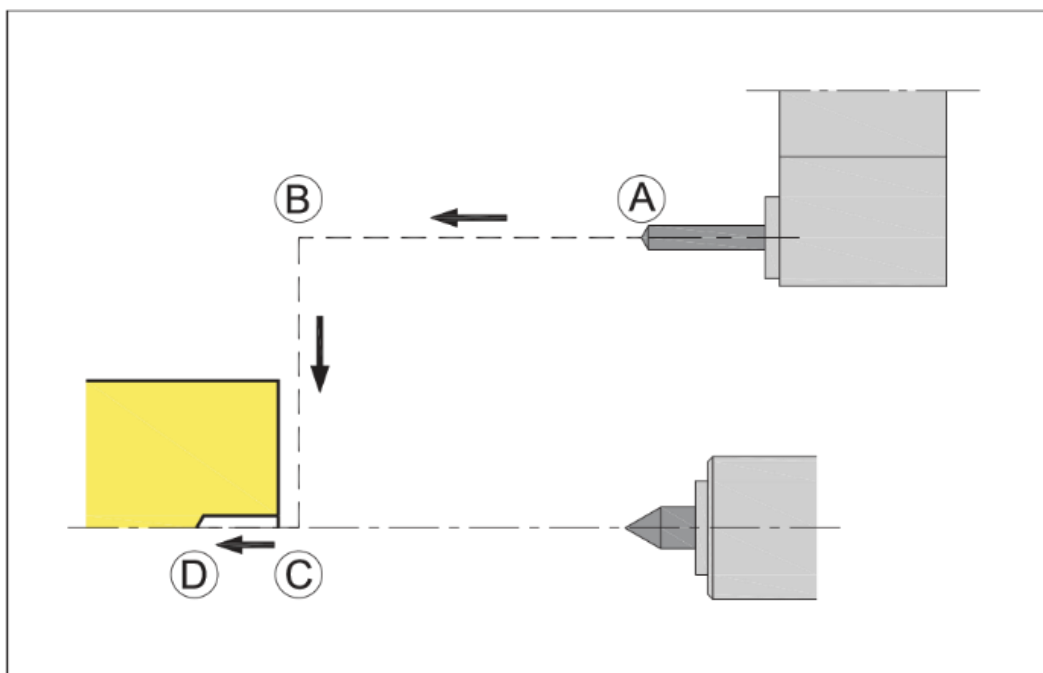


Figure 20-5

Typical example of a reversed rapid motion on a CNC lathe, used to bypass obstacles, for example, a tailstock

As Figure 20-5 shows, rather than programming a direct motion from the turret position to the cutting position (which would be directly from point A to point C), the tool motion was split. The approach towards the part will be in the order of A to B to C, at a rapid rate. From point C to point D, the actual cutting takes place. When cutting is done, tool will be rapid in reverse order, back to

the starting position. Rapid motion will be from D to C to B to A. This is a necessary precaution to bypass a potential obstacle, for example, the tailstock.

C. TYPE OF MOTION & TIME COMPARISON

Technique of programming each axis separately in individual blocks of the program, is recommended only for the purpose of bypassing possible obstacles during the tool path - and strictly for safety. This method of programming requires a slightly longer cycle time than the simultaneous multi-axis rapid motion. To compare the difference, consider a three-axis rapid motion, such as a typical tool approach in milling.

As an example, an old machine has rapid rate is of only 394 in/min (10000 mm/min) for each axis. Motion takes place between coordinate location of X2.36 Y0.787 Z0.2 (start point) and X11.812 Y3.54 Z1.0 (end point).

RAPID POSITIONING

The required time for tool travel along each axis can be easily calculated (*based on the initial Figure 20-2*):

X-axis time:

$$((11.812 - 2.36) \cdot 60) / 394 = 1.439 \text{ sec.}$$

Y-axis time:

$$((3.54 - 0.787) \cdot 60) / 394 = 0.419 \text{ sec.}$$

Z-axis time:

$$((1.0 - 0.2) \cdot 60) / 394 = 0.122 \text{ sec.}$$

If all three axes are moved simultaneously, the total time for positioning is 1.439 seconds, which is the longest time required for any axis to reach the end point. The program block will be:

G00 X11.812 Y3.54 Z1.0

If this motion were to be separated into three individual program blocks, the total time would be the result of individual times added together:

$$1.439 + 0.419 + 0.122 = 1.980 \text{ seconds}$$

which is about 37.5% longer. The percentage will vary, depending on the rapid motion rate and the rapid travel length, measured along each machine axis. Program blocks will be written separately:

G00 X11.812

Y3.54 Z1.0

Note that the modality of G00 rapid motion command does not require its repetition in the subsequent blocks.

D. REDUCTION OF RAPID MOTION RATE

During part setup or while proving a new program on the machine, CNC operator has an option to select a *slower rapid traverse rate* than the maximum established by machine manufacturer. This adjustment is done by the means of a special *rapid override switch*, located on the control system panel. This switch has typically four selectable positions, depending on the machine brand and the type of control system - *Figure 20-6*.

The second, third and fourth positions on the rapid motion override switch are rated as the *percentage of the actual rapid rate* - 25%, 50%, 100% respectively. They are set by the machine manufacturer. The first setting, sometimes identified by F0 (or F1) is a rapid motion rate set

through a *control system parameter*. The F0 (F1) setting should always be slower than any other setting, typically less than the lowest setting of 25%, such as 10%, for example.

151

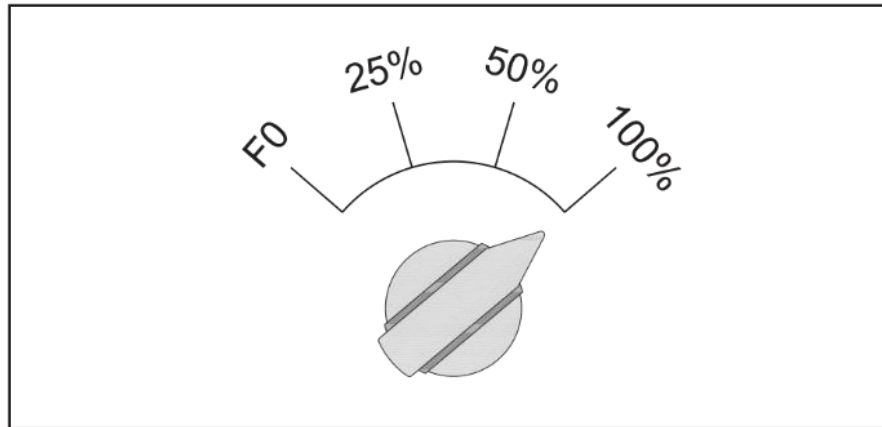


Figure 20-6

Rapid motion override switch set to 100% of rapid rate

Configuration of the rapid override switch varies between machines from different manufacturers. On some machines, the rapid motion may be stopped altogether, on others, tool will move at the slowest percentage and cannot be stopped with the override switch alone.

During actual production, after the program has been verified and optimized for best tool performance and productivity, the override switch should be set to the 100% pointer, to shorten overall cycle time.

E. RAPID MOTION FORMULAS

Calculations relating to the rapid tool motion can be expressed as formulas and used quickly at any time by substituting all known parameters. Relationships between rapid traverse rate, length of motion and the elapsed time can be expressed in the following three formulas:

$$T = \frac{L}{R} \times 60$$

$$R = \frac{L}{T} \times 60$$

$$L = \frac{T}{60} \times R$$

+ where ...

T = Required time in seconds

R = Rapid traverse rate per minute for the selected axis - *mm/min* or *in/min*
L = Length of motion - *mm* or *inches*

Units applied to the formulas must always be consistent within the selected system of measurement in the program. *Millimeters* and *millimeters per minute (mm/min)* must be used in the metric system. *Inches* and *inches per minute (in/min)* must be used with the imperial system. For any calculation relating to the rapid traverse time, the measuring units cannot be mixed.

F. APPROACH TO THE PART

Previous Figure 20-5 had an illustration of a safe tool approach as applied to a CNC lathe. For

CNC machining centers, the safety of part approach should be considered with equal care. Keep in mind that the general principles of rapid tool motion have to be considered for any machine. When approaching a part at a rapid rate, the cycle time can be somewhat shortened by keeping part clearances to the smallest safe minimum. Let's have a look at some potential problems.

In the following milling example (*Figure 20-7*), an approach to the part is made along the Z-axis, with a clearance of 0.05 inches (1.27 mm) in block N315:

N314 G90 G54 G00 X10.0 Y8.0 S1200 M03

N315 G43 Z0.05 H01

N316 G01 Z-1.5 F12.0

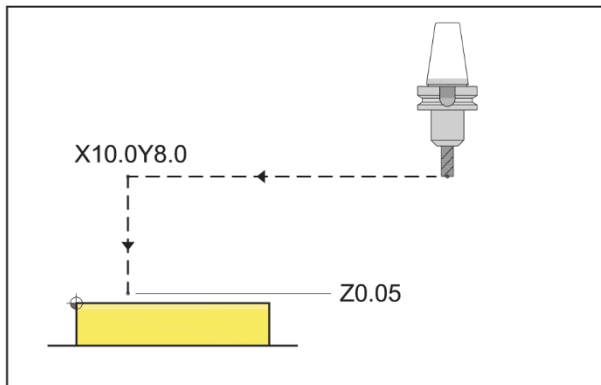


Figure 20-7

XY and Z axes approach to minimum clearance directly

There is nothing wrong with such a method of programming, providing the cutting tool is properly set and the part height is consistent from one part to another, as it should be. The clearance of 0.05 inches (1.27 mm) allows very little amount of unproductive cutting. On the other hand, an inexperienced CNC operator may not feel quite comfortable with such a small clearance, particularly during the early training stages. If the operator's convenience is considered as a significant factor contributing to the overall productivity, it might be a reasonable compromise to split the Z-axis motion into two separate motions (*Figure 20-8*):

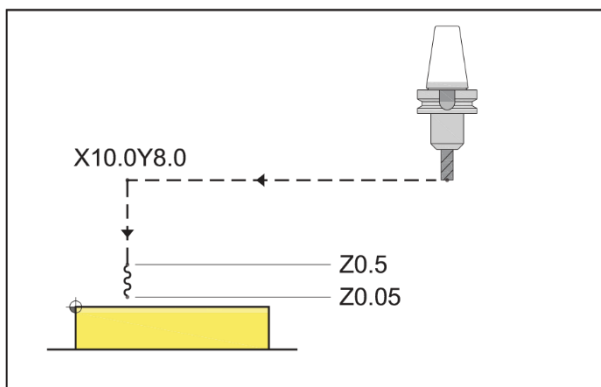


Figure 20-8

XY and Z axes approach to minimum clearance in a split motion

N314 G90 G54 G00 X10.0 Y8.0 S1200 M03

N315 G43 Z0.5 H01

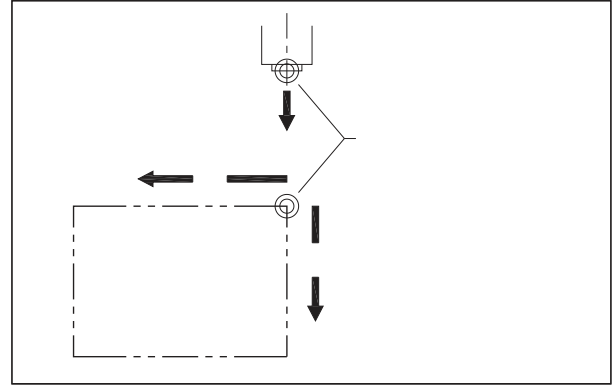
N316 G01 Z0.05 F100.0 N317 Z-1.5 F12.0

In this method, the rapid motion has been first programmed to a much more comfortable position of 0.500 inches above the part (N315). Then, the motion continued to the cutting start

point, using *linear* interpolation G01 in block N316. Since this is still a motion in the air, therefore not productive, a relatively heavy feed rate was used at the same time. As may be expected in such situations, there is a tradeoff.

Although the cutting time was slightly increased, at the same time, the CNC operator has been given an opportunity to use the feed rate override switch for testing the first part (used perhaps in a single block mode). Once the program is verified and debugged, the heavy feed rate in the non-cutting motion will speed up the operation and at the same time provide an extra safety clearance. The program with the split Z-axis motion can always be optimized later, although this may not be the best approach for repetitive jobs, since the setup is always ‘new’ for any repetition at a later date. However, it may be very useful when running lots of large numbers (several thousands, for example).

19. MACHINE ZERO RETURN



The ability of a control system to return a cutting tool from any position to the machine reference position is a critical feature of all modern CNC systems. Programmers and operators understand the term *machine reference position* as synonymous with the *home position* or *machine zero position*. This is *the position of all machine slides at one of the extreme travel limits of each axis*. Its exact position is determined by the machine manufacturer and is *not* normally changed during the machine working life. Return to that position is automatic, on request from the control panel, in MDI operation, or via a program.

A. MACHINE REFERENCE POSITION

The existence of machine reference position is for referencing purposes. In order that a CNC machine is a true precision machine tool, it requires more than just high-quality components, it requires some unique location that can be considered the machine origin point - a zero position - a home position. Machine reference position is exactly such a point.

Machine zero is a fixed position on a CNC machine that can be reached repeatedly, on request, through the control panel, MDI, or program code execution

Lathes

The machine reference position for two axis CNC lathes is logically no different from the reference position of machining centers. An easy access by CNC operator to the mounted part is the main determining factor. Both, the X and Z axes have their machine reference position at the furthest distance from the rotating part, which means away from the headstock area, consisting of the chuck, collet, face plate, etc.

For the X-axis, machine zero reference position is always at the extreme limit of travel *away from the spindle center line*. For the Z-axis, machine reference position is always at the extreme travel *away from the machine headstock*. In both cases, it normally means a positive direction towards the machine zero - the same as for machining centers. The illustration in *Figure 21-3* shows machine zero for a typical CNC lathe (rear type).

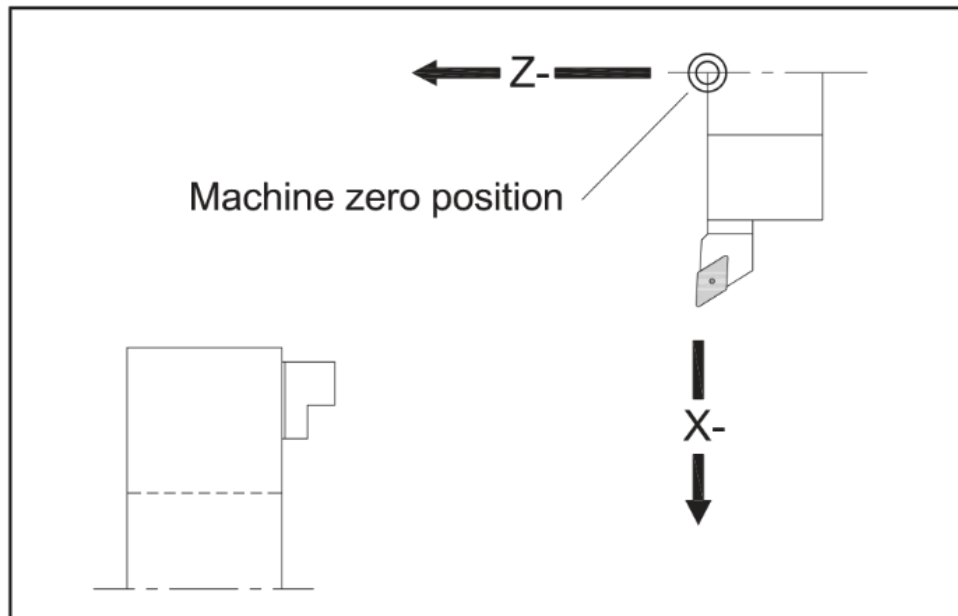


Figure 21-3

Machine zero position for a typical CNC lathe (rear type)

In the illustration, two arrows indicate tool motion direction towards the work area. Moving the tool from machine zero into the opposite direction will result in *overtravel* in the particular axis:

Tool motion from machine zero of a typical rear lathe:

X+ Z+ ... tool motion will overtravel

Setting the Machine Axes

From the earlier sections, remember that there is a direct relationship between the CNC machine, the cutting tool and the part itself. Work reference point (program zero or part zero) is always determined by the CNC programmer, the tool reference point is determined by the tool length at its cutting edge, also by the programmer.

Only the machine reference point (home position) is determined by the machine manufacturer and is located at a *fixed* position. This is a very important consideration.

Fixed machine zero means that all other references are dependent on this location

In order to physically reach machine reference position (home) and set the machine axes, for example, during part or fixture setup, there are three methods available to the CNC operator:

***Manually* - using the control panel of the system**

Machine operator will use the XYZ (machining centers) or the XZ (lathes) switches or buttons available for that purpose. One or more machine axes can be activated simultaneously, depending on the control unit

***Using the MDI* - Manual Data Input mode**

This method also uses the control panel. In this case, machine operator sets the MDI mode and actually programs the tool motion, using suitable program commands (G28, G30)

***In the CNC program* - during a cycle operation**

Using the same program commands as for MDI operation, CNC programmer, not the machine operator, includes machine zero return command (or commands) in the program, at desired places

When the CNC operator has performed the actual machine zero return, it is always a good idea to set the relative and absolute positions to zero on the display screen. Keep in mind that the relative

display can only be set to zero from the control panel and the absolute display can only be changed through a work offset, MDI mode, or the part program. This topic is normally a part of CNC machine operation training, directly at the machine.

For the last two methods of a machine zero return, CNC system offers four specific preparatory commands.

Program Commands

There are four preparatory commands relating to the machine zero reference position:

G27	Machine zero reference position return <i>check</i>
G28	Return <i>to</i> the <i>primary</i> machine zero reference position
G29	Return <i>from</i> the machine zero reference position
G30	Return <i>to</i> the <i>secondary</i> machine zero reference position (more than one is possible)

Of the four listed commands, G28 is used almost exclusively in two and three axis CNC programming. Its only purpose is to return the current tool to the machine zero position and do it along one or more axes specified in the G28 program block.

Command Group

All four preparatory commands G27 to G30 belong to the group 00 of the standard Fanuc designation that describes *non modal* or *one-shot* G-codes. In this designation, each G code of the 00 group must be repeated in every block it is used in. For example, when G28 command is used in one block for Z-axis and then it is used in the next block for X and Y axes, it has to be repeated in *each* block as needed:

N230 G28 Z.. (MACHINE ZERO RETURN Z-AXIS)

N231 G28 X.. Y.. (MACHINE ZERO RETURN XY AXES)

The G28 command in block N231 *must* be repeated. If this command is omitted, the *last motion* command programmed will be effective, for example, G00 or G01!

B. RETURN TO PRIMARY MACHINE ZERO

Any single CNC machine may have more than one machine zero reference point (home position), depending on its design. For example, many machining centers with a pallet changer have a *secondary* machine reference position, that is often used to align both the left and right pallets during pallet changing. The most common machine tool design is the one that uses only a single home position. To reach this primary home position, preparatory command G28 is used in the program and can also be used during MDI control operation.

G28 command moves the *specified* axis or axes to the home position, always at a rapid traverse rate. That means G00 command is assumed and does not have to be programmed. Axis or axes of the desired motion (with a value) must *always* be programmed. Only the programmed axes will be affected.

For example,

N67 G28

shows G28 programmed by itself in the block - this is an *incomplete* instruction. At least one axis must be specified with the G28 command, for example,

N67 G28 Y..

which will only send the Y-axis to machine zero reference position, or ...

N67 G28 Z..

will only send the Z-axis to machine zero reference position, and ...

N67 G28 X.. Y.. Z..

will send all three specified axes to machine zero reference position. Any multi-axis motion requires caution *watch for the infamous 'hockey stick' motion*.

Intermediate Point

One of the elementary requirements of programming is the alpha numerical composition of a word. In the program, every letter must be followed by one or more digits. The question is what *values* will the axes in G28 have? They will be the *intermediate* point for machine zero return motion. The concept of an intermediate motion in G28 or G30 is one of the most misunderstood programming features.

Commands G28 and G30 must always contain the *intermediate point* (tool position). By Fanuc design and definition, G28/G30 commands have a *built-in* motion to an intermediate point, *on the way* to machine zero. An analogy can be made to an airplane flight from Los Angeles, USA to Paris, France, that temporarily stops over in New York City. It may not be the most direct route, but it serves a certain specific purpose, for example, to refuel the aircraft.

Coordinate values of axes associated with G28 and

G30 commands always indicate an intermediate point

The purpose of the intermediate point, or position, is to shorten the program, normally by one block. This reduction is so marginal that the philosophy behind the design may be debated. Here is how the concept of intermediate point (position) works.

When G28 or G30 command is used in a program, *at least one axis must be specified* in the block. The value of that axis is the intermediate point, as interpreted by the control system. Absolute and incremental modes G90 and G91 make a great difference in interpretation of the G28 or G30 behavior and will be described shortly.

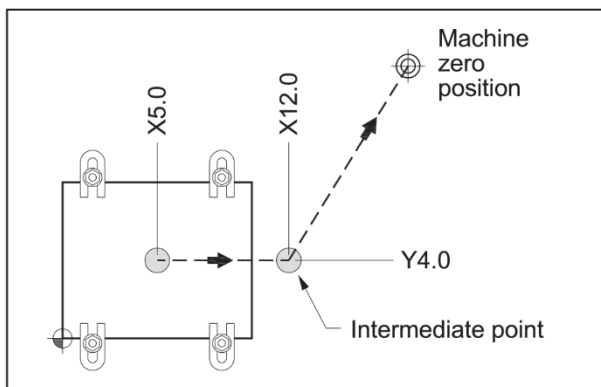


Figure 21-4

Intermediate point for machine zero return - XY axes shown

Tool motion in Figure 21-4 is from the central hole of the part. During such a motion, the tool can collide with the upper right clamp on its way to machine zero, if the motion to the home position were programmed directly. Only the X and Y axes are considered in the illustration. An *intermediate point* can be programmed in a safe location, without making the program any longer. Program *without* an effective intermediate point can be constructed as:

G90 ...

G00 X5.0 Y4.0 (MACHINED HOLE)

G28 X5.0 Y4.0 ... (MACHINE ZERO MOTION)

The same program *with* an actual intermediate point at a safe location will change slightly:

```
G90 ...  
G00 X5.0 Y4.0          (MACHINED HOLE)  
G28 X12.0 Y4.0 ...     (MACHINE ZERO MOTION)
```

Earlier examples have shown the reason behind this double motion. It is very simple - only to *save a single program block* - that is all. Its intended purpose is to use one block of program to achieve two motions, that would otherwise require two blocks. A safe program could also be:

```
G90 ...  
G00 X5.0 Y4.0          (MACHINED HOLE)  
X12.0                  (SAFE LOCATION)  
G28 X12.0 Y4.0 ...     (MACHINE ZERO RETURN)  
to produce the same final result, but with an extra block.
```

For example, using the intermediate position, a tool can be programmed to avoid an obstacle on the way to machine zero. If programmed with care, the intermediate position may be quite useful. Normally, it is more practical to make the intermediate motion equal to zero and move the cutting tool to machine zero directly. This is done by specifying the intermediate point as *identical* to the current tool position in *absolute* mode - *or* - by specifying a zero-tool motion in *incremental* mode.

Zero Return for CNC Lathes

For CNC lathe work, the G28 command may also be used, usually for setup. Common application of the machine zero return is also used, when at least one axis starts and ends at the machine zero position. This is quite often true of the X-axis but not of the Z-axis, which may be too far away on some larger lathe models.

Typically, a CNC lathe program will be designed in such a way, that machining of the first part will start from machine zero, but any subsequent part will be machined from a safe tool change position. This method is only practical if the program uses geometry offset, rather than the older G50 setting. The most common method of machine zero return on CNC lathes is the direct method, without an intermediate point, because no G91 is required, therefore, an error is more difficult to make:

```
N78 G28 U0
```

```
N79 G28 W0
```

These two blocks will return the cutting tool to machine zero in incremental mode, there is no intermediate motion applied. It is safer to move the X-axis first, using the incremental mode U, then the Z-axis, using the incremental mode W. If the work area is clear (watch for tailstock), both X and Z axes can be returned to machine zero at the same time:

```
N78 G28 U0 W0
```

Figure 21-6 illustrates a typical withdrawal of a boring bar from a hole, when the machining is completed.

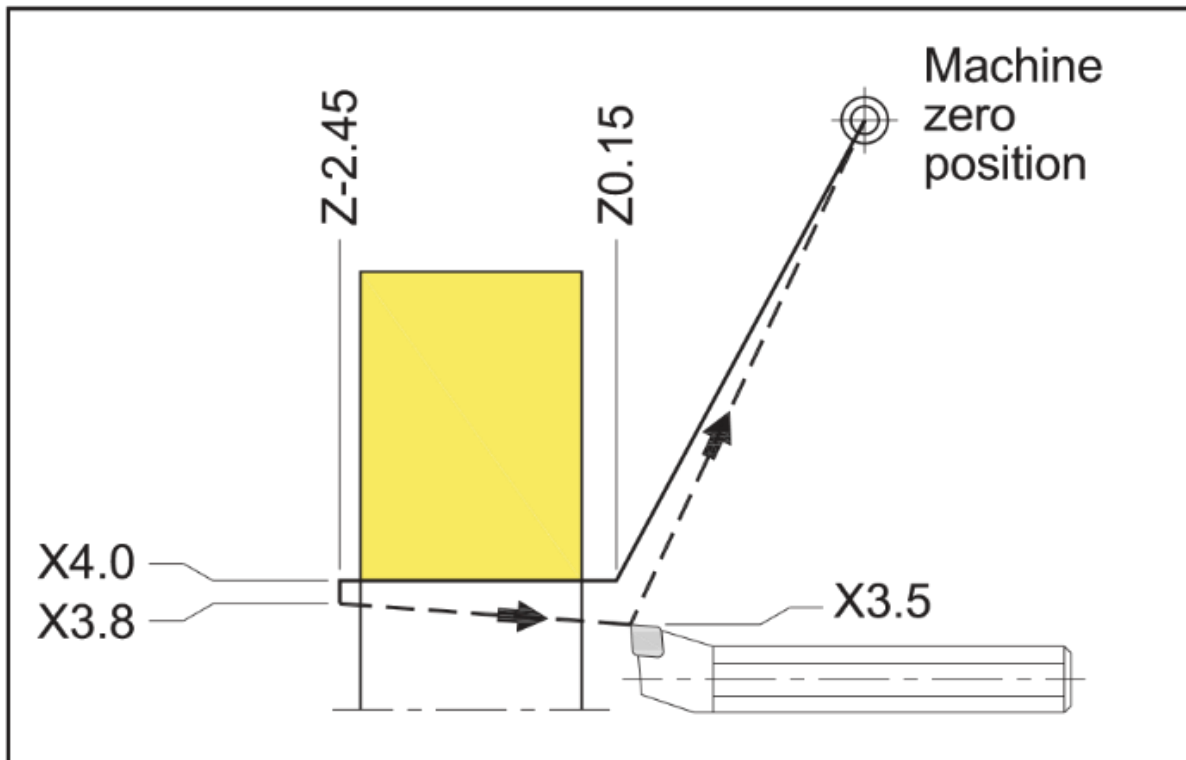


Figure 21-6

Machine zero return from a hole depth - turning application

When using position register command G50, the XZ setting must always be known for this command. In this case, programming rules for machine zero return are very similar. Assuming that the machine zero position is at the coordinate position X10.0 Z3.0, program for a boring tool can be written in two ways - one *without* using the G28 command, the other one *with* the G28 command.

Example 1:

The first example does not use G28 machine zero return command at all:

N1 G20 (EXAMPLE 1) ...

N58 G50 X10.0 Z3.0 S1000 (OLDER METHOD ONLY)

N59 G00 T0300 M42

N60 G96 S400 M03

N61 G00 G41 X4.0 Z0.15 T0303 M08

N62 G01 Z-2.45 F0.012

N63 X3.8 M09

N64 G00 G40 X3.5 Z0.15 M05

N65 X10.0 Z3.0 T0300 N66 M01

Example 2:

The second example will use G28 machine zero reference command, to achieve the same target position:

N1 G20 (EXAMPLE 2) ...

N58 G50 X10.0 Z3.0 S1000 (OLDER METHOD ONLY)

N59 G00 T0300 M42

N60 G96 S400 M03

N61 G00 G41 X4.0 Z0.15 T0303 M08

N62 G01 Z-2.45 F0.012

N63 G40 X3.8 M09

N64 G28 X3.5 Z0.15 M05 T0300 N65 M01

Most CNC programmers will likely feel more comfortable with the first example and saving one program block program will not likely be compelling enough to change their programming style. The second example (*Example 2*) can be programmed in incremental mode as well, using the U and W addresses, but it would not be too practical.

C. RETURN POSITION CHECK COMMAND

Much less common preparatory command G27 performs a checking function - and nothing else. Its only purpose is to check (which means to *confirm*), if the programmed position in the block containing G27 is at the machine zero reference point or not. If it is, the control panel indicator light for each axis that has reached the position will go on. If the reached position is not at the machine zero, program processing is interrupted by an error condition displayed on the screen as an alarm.

If the tool starting position is programmed at the machine zero reference (home), it is a good practice to return there as well, when machining with that cutting tool is fully completed. This is quite commonly done for CNC lathes, where the tool change (indexing) normally takes place in the same position, although this position does not always have to be machine zero. Usually, it is a safe position near the machined part.

Format for G27 command is:

G27 X.. Y.. Z..

where at least one axis must be specified.

When used in a program, the cutting tool will automatically rapid (no G00 necessary) to the position as specified by the axes in G27 block. The motion can be either in absolute or incremental mode. Note that no G28 command is used in the program.

N1 G20

N2 G50 X7.85 Z2.0 (OLDER METHOD ONLY)

N3 G00 T0400 M42

N4 G96 S350 M03

N5 G00 G42 X4.125 Z0.1 T0404 M08

N6 G01 Z-1.75 F0.012

N7 U0.2 F0.04

N8 G27 G40 X7.85 Z2.0 T0400 M09 N9 M01

In the example, block N8 contains G27, but no G00 or G28. This block instructs the CNC machine to return to position X7.85 Z2.0 and check, upon arrival to this target position, if that position is the machine zero in *all* specified axes (two axes in the example). Confirmation light will turn on, if the machine zero position has been reached. If the position is not confirmed, program will not proceed any further until the cause (misposition) is eliminated.

Compare the starting position in block N2 and the return position in block N8. Assuming that this position is at machine zero reference point in both X and Z axes, the above example will confirm OK position in the N8 block. Now, suppose that a small error has been made while writing

block N8, and the X amount was entered as X7.58 rather than the expected X7.85:

N8 G27 G40 X7.58 Z2.0 T0400 M09

In this case, the control system will return an error condition. The error is displayed automatically on the control screen (as an alarm). Control system will *not* process the remainder of the program, until any error is corrected. The light indicating *Cycle Start* condition will turn off and the problem *source* has to be found. When looking for source of the problem, always check *both positions*, the start position block, as well as the end position block. The error is quite easy to make in either block. Also note that any axis *not* specified in the block will not be checked for its actual (current) position.

Another important point is the cancellation of cutter radius offset and tool offset. G27 preparatory command should always be programmed with G40 command and Txx00 in effect (G49 or H00 for milling). If the tool offset or cutter radius offset is still in effect, the checking cannot be done properly, because the tool reference point is displaced by the offset value.

Here is how the first program (*Example 1*) listed earlier, can be modified to accept G27 command. Note that G27 will only move to the coordinates specified, *not* to any intermediate or other point. Block N65 will become the actual check block. The control system will move the machine axes to X10.0 Y3.0 and checks (confirms) whether this position is in fact the machine zero reference point. This is the reason *Example 1* could be modified, but not the second *Example 2*.

N1 G20

...

N58 G50 X10.0 Z3.0 S1000 (OLDER METHOD ONLY)

N59 G00 T0300 M42

N60 G96 S400 M03

N61 G00 G41 X4.0 Z0.15 T0303 M08

N62 G01 Z-2.45 F0.012

N63 X3.0 M09

N64 G00 G40 X3.5 Z0.15 M05

N65 G27 X10.0 Z3.0 T0300

N66 M01

Machine reference point return check can be done in either absolute or incremental mode. Absolute coordinates in block N65 (in the last example) can be replaced with incremental version:

N65 G27 U6.5 W2.85 T0300

There is a drawback to this command. A small price to pay when using this checking command is a slight cycle time loss. Because deceleration of a tool motion is built into the command by the control system, about one to three seconds may be lost when G27 command is implemented. This may be a significant loss if a large number of tools use G27 check in every program.

G27 command is seldom used with *geometry offset* setting of the tools, which is the current modern method. The G50 command is older and not used anymore on the newest CNC lathes, but many lathes are still used in industry that do need the G50 setting.

D. RETURN FROM MACHINE ZERO POINT

Preparatory command G29 is the exact opposite of G28 or G30 command. While G28 will automatically return the cutting tool to machine zero position, G29 command will return the tool

to its *original* position - also via an *intermediate point*.

In normal programming usage, the command G29 usually follows G28 or G30 command. Rules relating to the absolute and incremental axis designation are valid for G29 in exactly the same respect as to G28 and G30. All programmed axes are moved at the rapid traverse rate to the intermediate position first, defined by the preceding G28 or G30 command block. An example for a lathe application illustrates the concept:

(LATHE EXAMPLE) ...

T0303

...

G28 U5.0 W3.0

G29 U-4.0 W2.375

G29 command should always be issued in the canceled mode of both cutter radius offset (G40) and fixed cycles (G80), if either is employed in the program. Use the standard cancellation G-codes - G40 to cancel cutter radius offset and G80 to cancel a fixed cycle, before G29 command is issued in the program.

A schematic sketch of the tool motion is illustrated in *Figure 21-7*.

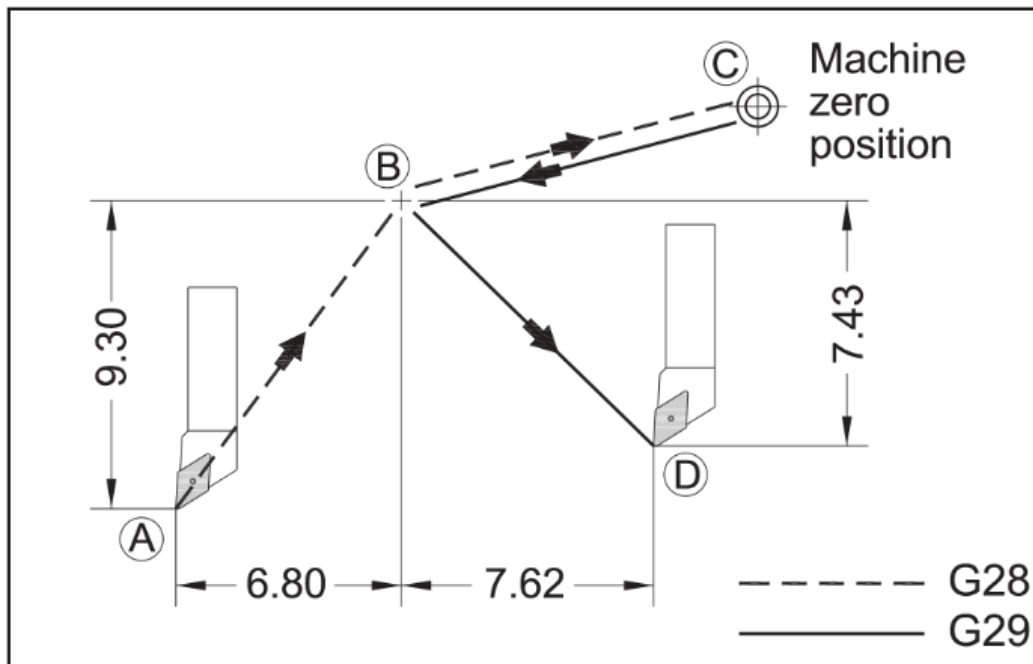


Figure 21-7

Automatic return from machine zero position

The illustration shows a tool motion from point A to point B first, then to point C, back to point B, and finally, to the point D. Point A is the starting point of the motion, point B is the intermediate point, point C is the machine zero reference point, and point D is the final point to reach - the actual target position.

Equivalent program commands, starting at the current tool position, which is point A, and resulting in the A to B to C to B to D tool path are quite simple:

G28 U18.6 W6.8

...

G29 U-14.86 W7.62

Of course, there would be some appropriate action programmed between the two blocks, for example, a tool change or some other machine activity.

Similar to G27 command, there is only a weak support for G29 among CNC programmers. It is one of the commands that can be very useful in some very rare cases, but virtually unnecessary for everyday work. However, it is always an advantage to know what 'tools of trade' are available in CNC programming. They may come handy.

E. RETURN TO SECONDARY MACHINE ZERO

In addition to the G28 machine zero command, specific CNC machines also have the G30 command. In this chapter, and the digital book generally, many examples apply equally to G28 and G30 commands and were sometimes identified as G28/G30 to cover both. So, what is different in G30 and why is this command needed in the first place?

By definition, G30 preparatory command is a machine zero return command to the *secondary machine zero position*. That position must be available on the machine at the time of purchase. Note the descriptive word is *secondary*, not *second*. In virtually all respects, G30 is identical to the G28, except that it refers to a *secondary* program zero.

This secondary program zero can be the physical second, third, or even fourth reference point, as specified by the machine manufacturer. Not every CNC machine has a secondary machine zero reference position, and not every CNC machine even needs one. This secondary machine reference point serves only some very special purposes, mainly for horizontal machining centers, pallets, and similar equipment.

Programming format for G30 command is similar to the G28 command, with an addition of the P-address:

G30 P.. X.. Y.. Z..

+ where ...

G30 = Indicates the selection of a secondary reference position

P = Can be P2, P3 and P4 to identify the secondary position (2-4)

XYZ = Is the intermediate point definition

(One axis minimum must be specified)

The most common use of a secondary machine zero reference point in CNC programming is for pallet changing or multi spindle machines. In the control unit parameter setting, the distance of the secondary reference point is set from the primary reference point and is not normally changed during the working life of the machine and the pallet changer (or another machine feature).

To distinguish between multiple secondary machine zero positions, address P is added in the G30 block (there is no P address used for G28). If the CNC machine has only a single secondary machine reference position, the address P is usually *not* required in the program, and P1 is assumed in such a case:

G30 X.. Y..

is the same as

G30 P1 X.. Y..

In this case, setting of the second reference point is within preset parameters of the control system. In respect to other programming considerations, G30 command is used in exactly the same way as the much more common G28 machine zero return command.

20. LINEAR INTERPOLATION

Linear interpolation is closely related to rapid positioning motion. While rapid tool motion is meant to be used from one position of the work area to another position *without cutting*, linear interpolation mode is designed for actual *material removal*, such as contouring, pocketing, face milling and many other cutting motions.

Linear interpolation is used in part programming to make a straight cutting motion from the cutter start position to its end position. It always uses the *shortest distance* a cutting tool path can take. The motion programmed in linear interpolation mode is always a straight line, connecting the contour start and end points. In this mode, the cutter moves from one position to another by the shortest distance between the end points. This is a very important programming feature, used mainly in contouring and profiling. Any angular motion (such as chamfers, bevels, angles, tapers, etc.) must be programmed in this mode to be accurate. Three types of motion can be generated in the linear interpolation mode:

Angular motion ... multiple axes

The term *linear interpolation* means that the control sys-

G01	Linear interpolation (linear motion)

In G01 mode, the feed rate function F must be in effect. The first program block that starts linear interpolation mode must have a feed rate in effect, otherwise an alarm will occur during the first run, after power on. Command G01 and feed rate F are modal, which means they may be omitted in all subsequent linear interpolation blocks, once they have been designated, and providing the feed rate remains unchanged. Only a change of coordinate location is required for the axis designation in a program block. In addition to a single axis motion, linear motion along two or three axes may be also programmed simultaneously.

A. START AND END OF THE LINEAR MOTION

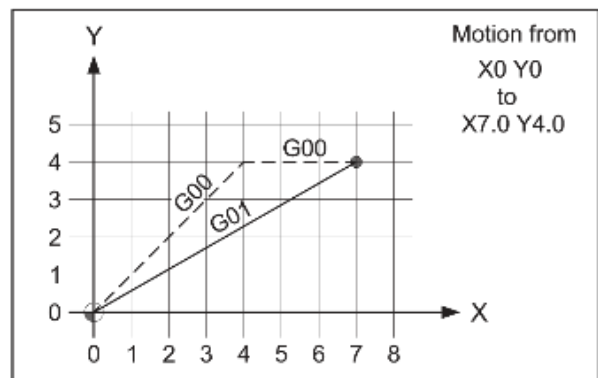
Linear motion, like any other motion in CNC programming, is a motion between two end points of a contour. It has the *start* position and the *end* position. Any *start* position is often called the *departure* position, the *end* position is often called the *target* position. Start of a linear motion is defined by the current tool position, its end is defined by the target coordinates of the current block. It is easy to see that the end position of one motion will become the start position of the next motion, as the cutter moves along the part, through all contour change points.

B. SINGLE AXIS LINEAR INTERPOLATION

- Horizontal motion ... single axis only
- Vertical motion ... single axis only
-

tem is capable to calculate thousands of intermediate coordinate points between the start point and end point of the cut. Result of this calculation is the shortest path between the two points. All calculations are automatic - the control system constantly coordinates and adjusts the feedrate for all cutting axes, normally two or three.

LINEAR COMMAND



Programmed tool motion along any single axis is always a motion parallel to that axis, regardless of the motion mode. Programming in either G00 or G01 mode will result in the same programmed end point, but at different feed rates and with different results. Evaluate Figure 22-1 for comparison of the two motion modes.

Figure 22-1

Comparison of the rapid mode and the linear interpolation mode

For CNC machining centers and the related machines, all tool motions that are parallel to the table edges are single axis motions. On CNC lathes, many external and internal operations, such as facing, shoulder turning, diameter turning, drilling, tapping and others, are programmed as single axis motions. In all cases, a single axis motion can be along either the *vertical* or the *horizontal* axis, within the current (working) plane. A single axis motion can never be an angular motion, which requires two or three axes. Another name for a motion that is parallel to a machine axis is *orthogonal* - horizontal or vertical only.

163

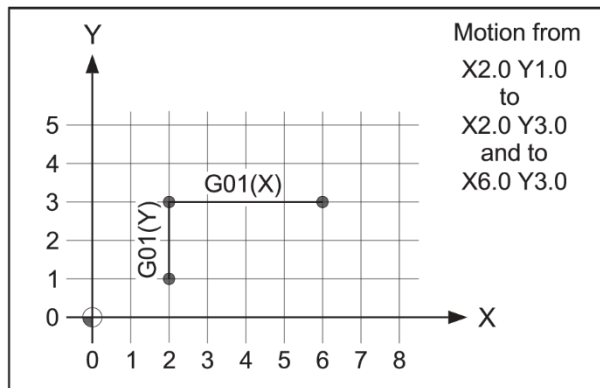


Figure 22-2

Single axis linear interpolation motion

Figure 22-2 illustrates a single axis linear interpolation motion, one along X-axis and the other along Y- axis.

C. TWO AXES LINEAR INTERPOLATION

Linear motion can also be programmed along *two axes* simultaneously. This is a very common situation when the start point of linear motion and its end point have at least two coordinates that are different from each other, while in linear interpolation mode G01. The result of this two-axis motion is a straight tool motion *at an angle*. Such motion will always be the shortest distance between the start point and end point and results in a straight line at an angle calculated by the control - Figure 22-3.

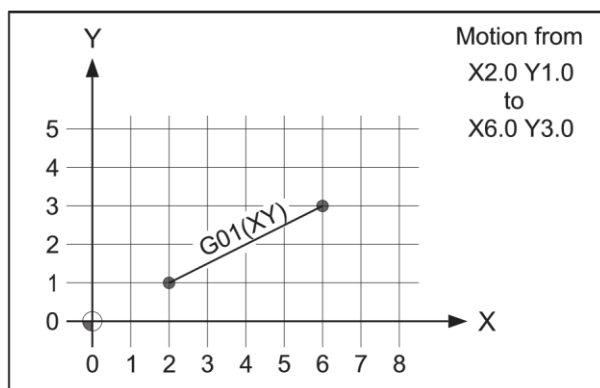


Figure 22-3

Two axes simultaneous linear interpolation motion

D. THREE AXIS LINEAR INTERPOLATION

Linear motion that takes place along three axes at the same time, is called the *three-axis linear interpolation*. A simultaneous linear motion along three axes is possible on virtually all CNC machining centers. Programming a linear motion of this kind is not always easy, particularly when working with complex parts. Due to many difficult calculations involved in this type of tool motion, the manual programming method is not efficient enough. Such programming projects more than justify an investment into a professional computer-based programming system, such as the very powerful and widely used *Mastercam™*, that is based on modern computer technology combined with machining know-how. This type of programming is using desktop computers and is affordable by virtually all machine shops. Computer based programming is not a subject of this digital book, but its general concepts are discussed briefly in the last chapter of the digital book.

Three-axis (XYZ) simultaneous linear motion is illustrated in *Figure 22-4*.

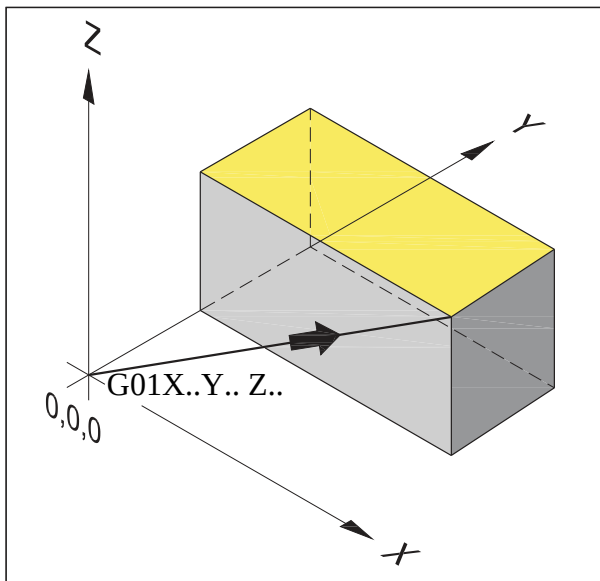


Figure 22-4

Three axes simultaneous linear interpolation motion

E. PROGRAMMING FORMAT

In order to program a tool motion in linear interpolation mode, use preparatory command G01 along with one, two, or three axes of tool motion, as well as a cutting feed rate (F address) suitable for the job at hand:

G01 X.. Y.. Z.. F..

All entries in linear motion block are modal and need to be programmed only if they are new or changed. Only the block instruction (word) that is affected by the change needs to be included in the program block.

Depending on which programming method is selected, linear interpolation motion may be programmed in absolute or incremental mode, using G90 and G91 preparatory commands for milling, and incremental addresses U and W for turning.

LINEAR INTERPOLATION

F. LINEAR FEED RATE

The actual cutting feed rate for a defined tool motion can be programmed in two modes:

... per time mm/min or in/min

... per spindle revolution mm/rev or in/rev

The selection depends on the machine type and dimensional units used. Typically, CNC machining centers, drills, mills, routers, flame cutters, laser profilers, wire EDM, etc., use *feed rate per time*. CNC lathes and turning centers typically use *feed rate per revolution*.

Feed rate Range

Every CNC system supports cutting feed rate only within a certain range. For linear interpolation in milling applications, the typical lowest feed rate is 0.0001, either as *in/min*, *mm/min* or *deg/min*. The lowest feed rate for linear interpolation in turning is dependent on the minimum increment of the coordinate axes XZ. The following two tables point out typical ranges a normal CNC system can support. The first table is for milling, the second table is for turning. All units used in part programming are represented.

Minimum motion increment	MILLING
0.001 mm	0.0001 - 240000.00 mm/min
0.001 degree	0.0001 - 240000.00 deg/min
0.0001 inch	0.0001 - 240000.00 in/min
Minimum motion increment	TURNING
0.001 mm	0.00001 - 500.00000 mm/min
0.001 degree	0.00001 - 500.00000 deg/min
0.0001 inch	0.000001 - 50.000000 in/min

It may appear that the maximum feed rate that can be used is unusually high. For actual cutting, that is true. However, keep in mind that these ranges are relative to the control system, *not* to the machine. Machine manufacturer will always limit the maximum feed rate, according to the machine design and its capabilities. Control system only provides the *theoretical range*, that is more for the benefit of machine designers than the actual user. The intent in this case is to allow machine manufacturers flexibility within current technological advances. As technology changes, the control system manufacturers will have to respond to these changes as well, by increasing the available ranges.

Individual Axis Feed rate

The subject of actual cutting feed rate per axis is not crucial in programming at all. It is included here for the mathematically oriented and interested readers only. There is no need to know the following calculations at all - the CNC system will do them every time, all the time, accurately and automatically. On the other hand, here it is anyway.

In order to keep linear motion as the *shortest* motion between two points, the CNC unit must always calculate the feed rate for *each axis individually*. Depending on the direction of linear motion (its angular value), the computer will '*speed up*' one axis and '*hold back*' the other axis *at the same time*, and it will do it constantly during every cut. The result is a straight line between the start and end points of the linear contour. Strictly speaking, it is not a straight line but a jagged line, with edges so diminutive in size that they are virtually impossible to see, even under magnification. For all practical purposes, the result *is* a straight line.

Calculations are done by the CNC system, according to the following entries, as illustrated in Figure 22-5.

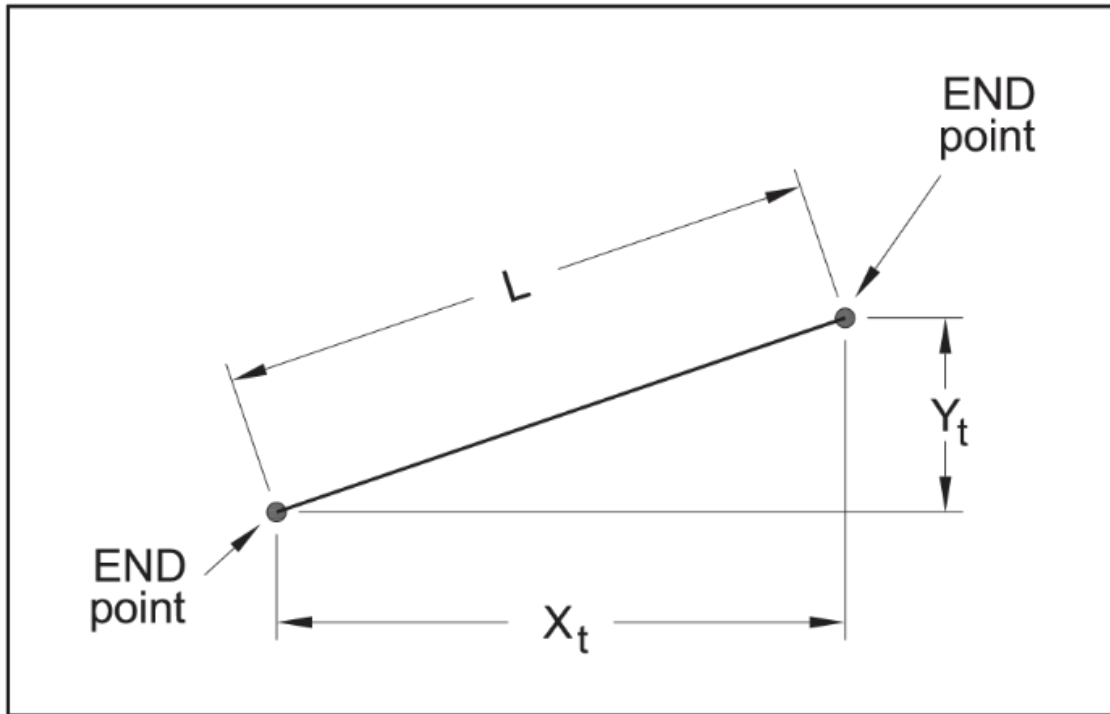


Figure 22-5

Data for the calculation of individual axis linear feed rate

Evaluate the following example of linear motion and try to apply the formulas listed afterwards:

G00 X10.0 Y6.0 (START POINT)

G01 X14.5 Y7.25 F12.0 (END POINT)

This linear motion takes place between two end points, from the starting point at X10.0 Y6.0 to the end point at X14.5 Y7.25 - feed rate is programmed at 12.0 in/min as F12.0. That means the actual travel motion along each axis is either known or it can be calculated:

$$X_t = 14.5 - 10.0 = 4.5$$

$$Y_t = 7.25 - 6.0 = 1.25 \quad Z_t = 0$$

The length L of tool total motion (as illustrated) is the actual compound motion, and can be calculated by using the well-known *Pythagorean Theorem* (see page 506):

$$L = \sqrt{X_t^2 + Y_t^2 + Z_t^2}$$

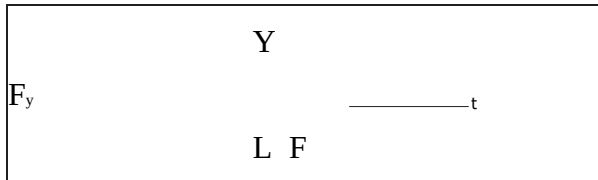
The above formula is quite common, based on the square root of the total sum of squares of sides, that will result in the value of 4.6703854 as travel length in the example:

$$L = \sqrt{4.5^2 + 1.25^2 + 0^2} = 4.6703854$$

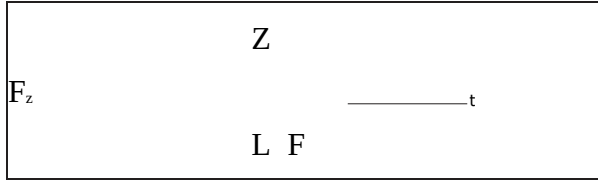
Control system will *internally* apply the formulas and calculate the actual motion along X-axis (4.25), as well as along Y-axis (1.25), plus length of the motion itself, which is 4.6703854. From these values, the computer system will calculate the X and Y axis feed rate - there is no motion that takes place along the Z-axis:

$$F_x = \frac{L}{F} \quad X \quad t$$

$$F_x = 4.5 / 4.6703854 \ 12 = 11.562215$$



$$F_x = 1.25 / 4.6703854 \ 12 = 3.2117263$$



$$F_x = 0 / 4.6703854 \ 12 = 0.0$$

In this example, there is no Z-axis motion. If Z-axis were part of the tool motion, for example, during a simultaneous three-dimensional linear motion, the procedure will be logically identical, with the inclusion of Z-axis in the calculations.

G. PROGRAMMING EXAMPLE

In order to illustrate the practical use of linear interpolation mode in a CNC program, here is a simple example, shown in *Figure 22-6*.

For even more comprehensive understanding, the example will be presented twice. One tool motion will start and end at the P1 location and will be programmed in the order of P1-P2-P3-P4-P5-P6-P7-P8-P1. The other program example will start at the same P1 location but will continue in the opposite direction P1-P8-P7-P6-P5-P4-P3-P2-P1.

Chapter 22

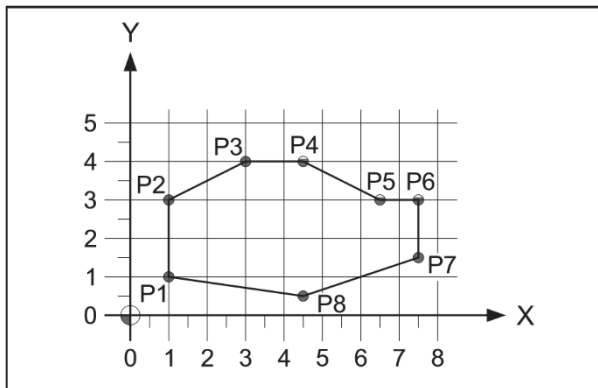


Figure 22-6

Example illustration for a simple linear interpolation

Example 1: ('CLOCKWISE' DIRECTION FROM P1)

G90 ...	(ABSOLUTE MODE)
G01 X1.0 Y3.0 F..	(P1 TO P2)
X3.0 Y4.0	(P2 TO P3)
X4.5	(P3 TO P4)
X6.5 Y3.0	(P4 TO P5)
X7.5	(P5 TO P6)

Y1.5	(P6 TO P7)
X4.5 Y0.5	(P7 TO P8)
X1.0 Y1.0 ...	(P8 TO P1)

Example 2:

('COUNTERCLOCKWISE' DIRECTION FROM P1)

G90 ...	(ABSOLUTE MODE)
G01 X4.5 Y0.5 F..	(P1 TO P8)
X7.5 Y1.5	(P8 TO P7)
Y3.0	(P7 TO P6)
X6.5	(P6 TO P5)
X4.5 Y4.0	(P5 TO P4)
X3.0	(P4 TO P3)
X1.0 Y3.0	(P3 TO P2)
Y1.0 ...	(P2 TO P1)

Linear interpolation provides means of programming all orthogonal (*i.e.*, vertical and horizontal) motions, as well as angular tool motions as the shortest linear distance between two points. Cutting feed rate must be programmed in this mode, for proper metal removal. Note that the coordinate location that has *not* changed from one point to the next one block to the next - is not repeated in the subsequent block or blocks.

21. BLOCK SKIP FUNCTION

In many control and machine manuals, block skip function is also called the *block delete* function. The expression '*block delete*' offers rather a misleading description, since no program blocks will actually be deleted during processing, but only *skipped*. For this good reason, the more accurate description of this function is the *block skip* function, a term used in the digital book. This function is a standard feature of virtually all CNC controls. Its main purpose is to offer the programmer some additional flexibility in designing a program for no more than *two conflicting possibilities*. In the absence of a block skip function, the only alternative is to develop two individual part programs, each covering one unique possibility.

A. TYPICAL APPLICATIONS

To understand the idea of two conflicting possibilities, consider this programming application. The assignment is to write a program for a facing cut. The problem is that the blank material for parts delivered to the CNC machine is not consistent in size. Some blanks are slightly smaller in size and can be faced with a single cut. Others are larger and will require two facing cuts. This is not an uncommon occurrence in CNC shops and is not always handled efficiently. Making two inefficient programs is always an option, but a single program that covers both options is a better choice - but only if the *block skip function* is used in such a program and machine operator understands it.

This challenge illustrates a situation, where two conflicting options are required in a program at the *same* time. The most obvious solution would be to prepare two separate programs, each properly identified as to its purpose. Such a task can be done quite easily, but it will be a tedious, time consuming and an inefficient process. The only other solution is to write a *single* program, with tool motions covering facing cuts for *both* possibilities. To avoid air cutting for those parts that require only one cut, a block skip function will be provided in the program and applied to all blocks relating to the *first* facing cut. The 'second' cut will always be required!

Other common applications of the block skip function include a selective ON/OFF status toggle, such as the coolant function, optional program stop, program reset, etc. Also useful are applications for bypassing a certain program operation, applying or not applying a selected tool to a part contour and others. Any programming decision that requires a choice from *two* predetermined options is a good candidate for the block skip function.

B. BLOCK SKIP SYMBOL

To identify block skip function in a program, a special programming symbol is required. This block skip function symbol is represented by a forward slash [/]. Control system will recognize the slash as a code for block skip. For most of CNC programming applications, the slash symbol is placed as the *first* character in a block:

Example 1:

```
N1 ... (ALWAYS PROCESSED)
N2 ... (ALWAYS PROCESSED)
N3 ... (ALWAYS PROCESSED)
/ N4 ... (PROCESSED ONLY IF BLOCK SKIP IS OFF)
/ N5 ... (PROCESSED ONLY IF BLOCK SKIP IS OFF)
/ N6 ... (PROCESSED ONLY IF BLOCK SKIP IS OFF)
N7 ... (ALWAYS PROCESSED)
N8 ... (ALWAYS PROCESSED)
```

On *some* control systems, the block skip code can also be used selectively for certain addresses *within* a block, rather than at its beginning. Check control manual if such a technique can be used - it can be very powerful:

Example 2:

N6 ...

N7 G00 X50.0 / M08

N8 G01 ...

...

In those cases, when the control system does allow the block skip *within* a programmed block, all instructions *before* the slash code will be executed, regardless of the block skip toggle setting. If the block skip function is turned ON (block skip function is active), only instructions *following* the slash code, will be skipped. In the *Example 2*, coolant function M08 (block N7) will be skipped. If the block skip function is turned OFF (block skip function is not active), the whole block will be executed in *Example 2*, including the coolant function.

C. CONTROL UNIT SETTING

Regardless of the slash code position within a block, the program will be processed in two ways. Either in its entirety, or any instruction following the slash within a block will be skipped (ignored). The final decision whether or not to use block skip function is made during actual machining,

by the operator, depending on the machining requirements. For this purpose, a push button key, a toggle switch, or a menu item selection is provided on the control panel of the CNC unit. Selection of block skip mode can be either as *active* (ON) - or *inactive* (OFF).

Most programs will not require any block skip functions. In such cases, the setting mode for block skip function on the control panel is irrelevant, but OFF mode is strongly recommended. The switch setting becomes *very* important, if the program contains even a *single* block containing the slash symbol. The active setting ON will cause all instructions in a block following the slash code to be ignored during program processing. The inactive setting OFF will cause the control to ignore the slash code and process all instructions written in the program block.

Block skip function set to ON position means

'Ignore all block instructions following the slash'

Block skip function set to OFF position means

'Process all block instructions'

In the *Example 1* listed earlier, the contents of blocks N4, N5 and N6 will be ignored, if the block skip function is ON. They will be processed, if the switch setting is OFF. The *Example 2*, also listed earlier, contains a slash in block N7. The slash symbol precedes miscellaneous function M08 (coolant ON). If the skip function switch is ON, the coolant will be ignored; if it is OFF, the coolant function will be effective. This application may be useful in a dry run mode, to bypass coolant flood during program verification, if no manual override is available.

Not all controls allow the slash code in any other block position, except as the first character in the block: / N..

D. BLOCK SKIP AND MODAL COMMANDS

To understand the way how modal values work with skipped blocks, recall that modal commands can be specified only once in a program, in the block where they occur first. Modal commands are not repeated in the subsequent blocks, as long as they remain unchanged.

In programs where the block skip function is not used at all, there is nothing to do. When the block skip function *is* used, watch carefully all *modal* commands. Remember that a command established in a block using the slash code will not always be in effect. It depends on the setting of block skip switch. Any modal command that has to be carried over from a section with slash codes to the section without slash codes may be lost if the block skip function is used. Overlooking modal commands when programming block skip function can result in a program with serious errors.

A very simple programming solution to avoid this potential problem is available. Just *repeat* all modal commands in the program section that will not be affected by the block skip function.

Compare the following two examples:

Example A - Modal commands *are not* repeated:

N5 G00 X10.0 Y5.0 Z2.0
/ N6 G01 Z0.1 F30.0 M08

— —
N7 Z-1.0 F12.0 (G01 AND M08 MISSING) N8 ...

Example B - Modal commands *are* repeated:

N5 G00 X10.0 Y5.0 Z2.0
/ N6 G01 Z0.1 F30.0 M08
N7 ~~G01~~ Z-1.0 F12.0 ~~M08~~ N8 ...

In both examples A and B, the program block containing slash code indicates intermediate Z-axis position as Z0.1. This position may be required only in certain cases during machining and the operator will decide whether to use it or not, and also when to use it.

The critical block, identified in the examples as N6, contains several modal functions. Commands G01, Z0.1, F30.0 and M08 will all remain in effect, unless they are canceled or changed in any following block. From block N7 it is apparent that only the Z-coordinate position and the cutting feed rate value have changed. However, the G01 and M08 commands are not repeated in the example A and will *not* be in effect, if the block skip switch is set ON.

Both examples A and B will produce identical results, but only if the block skip function is inactive (OFF). The control system will then execute the instructions in *all* blocks, in the order of programming sequence.

The processing result will be different for each programming example shown. If the block skip function is active (ON) - block instructions following the slash code will *not* be processed. The next example A yields an unacceptable result, with a fairly possible collision. Example B uses careful and thoughtful approach with very little extra work. These are the results when block N6 is skipped:

Example A - Modal commands *are not* repeated:

N5 G00 X10.0 Y5.0 Z2.0 (RAPID MOTION)
N7 Z-1.0 F12.0 (RAPID MOTION)
N8 ...

Example B - Modal commands *are* repeated:

N5 G00 X10.0 Y5.0 Z2.0 (RAPID MOTION)
N7 G01 Z-1.0 F12.0 M08 (FEED RATE MOTION) N8 ...

Note that the linear motion G01, feed rate F30.0 and coolant M08 are all skipped in the example

A. The X and Y axes have not been updated in either example and will remain unchanged. The conclusion is that example A will result in a Z-axis rapid motion in two consecutive blocks, causing a potentially dangerous situation. In the correct version, listed as example B, the programmed repetition of all commands - G01, F12.0 and M08 - assures the program will always be processed as intended. In the next section of this chapter, principles of program design for different practical applications will be presented.

In the summary, there is one basic rule for developing CNC programs with blocks using the block skip function:

Always program *all* instructions, even if it means repeating some program values and commands that have to be preserved

The slash symbol can be placed into the program *after* the program has been designed for *both* options. Just place the slash in those blocks that define the optional skip of all selected program blocks. *Always check program!*

Any CNC program containing block skip function should be checked at least twice - in ON and OFF mode

The result of this double check must be *always* satisfactory, whether processed with block skip in effect or without it. If an error is detected, *even a very minor error*, correct it first! After the correction, check the program at least *twice* again, covering both types of processing. Main reason for this special double check is that a correction made for one type of processing may cause (in some cases) a different error for the other type of processing.

E. PROGRAMMING EXAMPLES

Block skip function is very simple, often neglected, yet it is a powerful programming tool. Many programs can benefit from a creative use of this feature. Type of work and some thinking ingenuities are the only criteria for its successful implementation. In the following examples, some practical applications of the block skip function are shown. Use these examples as start points for general program design or when covering similar machining applications.

Variable Stock Removal

Removal of excessive stock material is typical during a rough cutting. When machining irregular shapes (castings, forgings, etc.) or rough facing on lathes, it may be difficult to determine the number of cuts. For example, some castings for a given job may have only the minimum excessive material, so one roughing or facing cut will be sufficient. Other castings for the same job may be larger and two roughing or facing cuts are needed.

If the program is designed in such a way that there is only *one* roughing or facing cut, problems may occur during machining of heavy stock. Programming *two* cuts for all parts produces a safer program but will be inefficient for parts with a minimum stock. There will be too many tool motions known as 'cutting air' when the stock is minimal.

Example - Variable stock face:

A face cutting of a stock that varies in size is a common problem in CNC work. A suitable solution is identical for turning and milling - the program should include tool motions for *two* cuts and the block skip function will be used on all blocks relating to the *first* cut.

Here is a lathe example of typical face cut, when the facing stock varies between 0.08 (2 mm) and 0.275 (7 mm). After considering several machining options, the programmer decides that the reasonable maximum stock that can be faced in a single cut will be 0.135 (3.5 mm) - *Figure 23-1*.

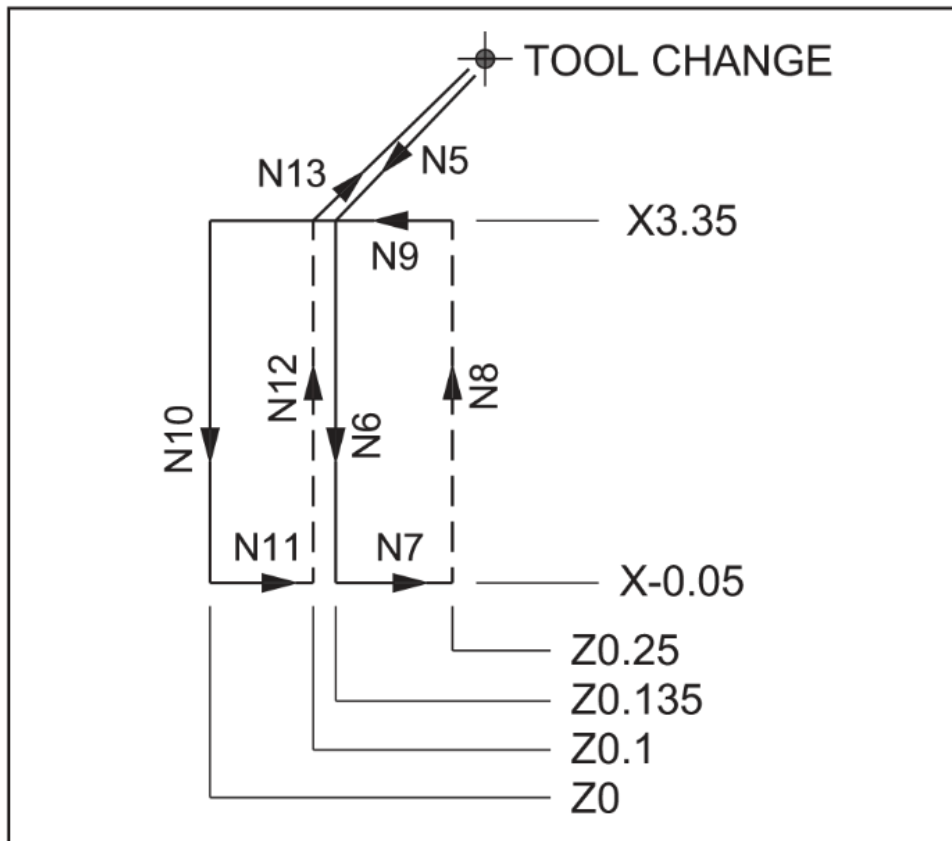


Figure 23-1

Variable stock for facing in a turning application - program O2301

O2301 (TURNING)

(VARIABLE FACE STOCK)

N1 G20 G40 G99

N2 G50 S2000

N3 G00 T0200 M42

N4 G96 S400 M03

N5 G41 X3.35 Z0.135 T0202 M08

/ N6 G01 X-0.05 F0.01

/ N7 G00 Z0.25

/ N8 X3.35

N9 G01 Z0 F0.05

N10 X-0.05 F0.01

N11 G00 Z0.1

N12 X3.5

N13 G40 X12.0 Z2.0 T0200

N14 M30 %

Block N5 contains the initial tool approach motion. The next three blocks are preceded by a slash. In N6, the tool cuts off the front face, at Z0.135; N7 moves the tool away from the face, block N8 is a rapid motion back to the initial diameter. There are no other blocks to be skipped after the block N8. N9 block contains a feed rate to the front face Z0, N10 is the front face cutting

motion, N11 is the clearance motion, followed by standard final blocks.

Evaluate the example not once but at least *twice* - it shows what exactly happens. During the *first* evaluation, read *all* blocks, including those with slash. During the *second* time, ignore all blocks containing the slash code. There will be identical results when compared with the first evaluation. The only difference will be the number of actual cuts - *one*, not two. In milling, the procedure is very similar.

An example for a milling application uses a 5-inch face mill. The excessive material stock to be faced varies between 0.120 and 0.315. The largest reasonable depth of cut selected will be 0.177 (4.5 mm) - *Figure 23-2*.

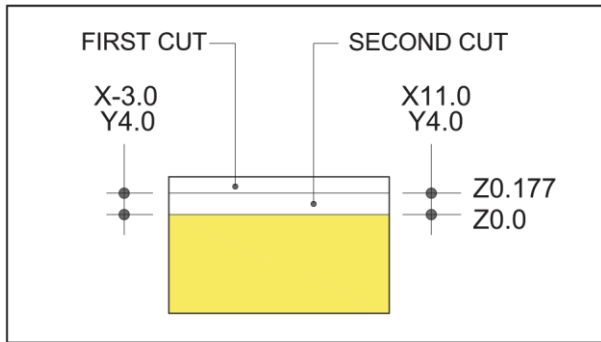


Figure 23-2

Variable stock for facing in a milling application - program O2302

O2302 (MILLING)

(VARIABLE FACE STOCK)

N1 G20

N2 G17 G40 G49 G80

N3 G90 G00 G54 X11.0 Y4.0 N4 G43 Z1.0 S550 M03 H01

N5 G01 Z0.177 F15.0 M08

/ N6 X-3.0 F18.0

/ N7 Z0.375

/ N8 G00 X11.0 N9 G01 Z0

N10 X-3.0 F18.0

N11 G00 Z1.0 M09

N12 G28 X-3.0 Y4.0 Z1.0

M13 M30

%

Block N5 in the example contains Z-axis approach to the first cut, at Z0.177 level. The next three blocks can be skipped if necessary. In N6 block, the face mill actually cuts at Z0.177 position, N7 is the tool clearance motion after the cut, and N8 returns the tool to the initial X-position. There are no other blocks to be skipped after block N8.

Block N9 does not need a feed rate for a good reason - it will be *either* F15.0 *or* F18.0, depending on whether blocks N6 to N8 were skipped or not. The feed rate is very important in block N10. Such repetition guarantees the required feed rate is in the critical block, when actual cutting takes place.

Both lathe and mill examples should offer at least some basic understanding of the *logic* used

in program development, using the block skip function. Exactly the same logical approach can be used for more than two cuts and can also be applied to operations other than face cutting.

Machining Pattern Change

Another application, where block skip function may be used efficiently, is a simple family programming. The term *family programming* means a programming situation where there may be a slight difference in design between two or more parts. Such a small variation between *similar* parts is often a good prospect for block skip function. A minor deviation in a machining pattern from one drawing to another can be adapted in a single program using the block skip function. Following two examples show typical possibilities of programming a *change of tool path*. In one example, the emphasis is on a skipped machining location. In the other, the emphasis is on the pattern change itself. Both examples are in metric and illustrate a simple grooving operation. For the lathe example, *Figure 23-3* is related to program O2303.

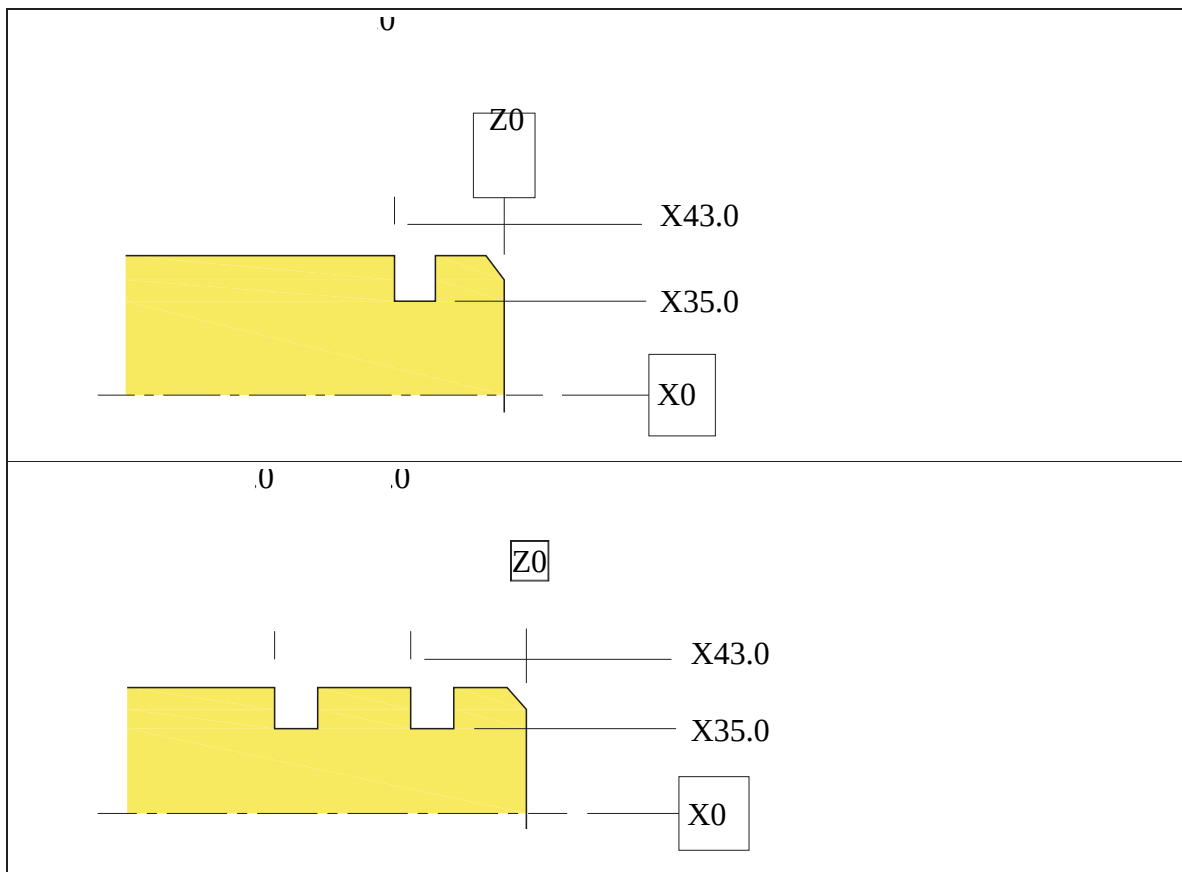


Figure 23-3

Variable machining pattern - turning application - O2303

The upper image shows machining result with block skip function set ON, the lower image shows machining result with block skip function set OFF, using the *same* program.

O2303 (LATHE EXAMPLE)

N1 G21

...

N12 G50 S1800

N13 G00 T0600 M42 N14 G96 S100 M03 N15 X43.0 Z-20.0 T0606 M08

N16 G01 X35.0 F0.13

N17 G00 X43.0

/ N18 Z-50.0

/ N19 G01 X35.0

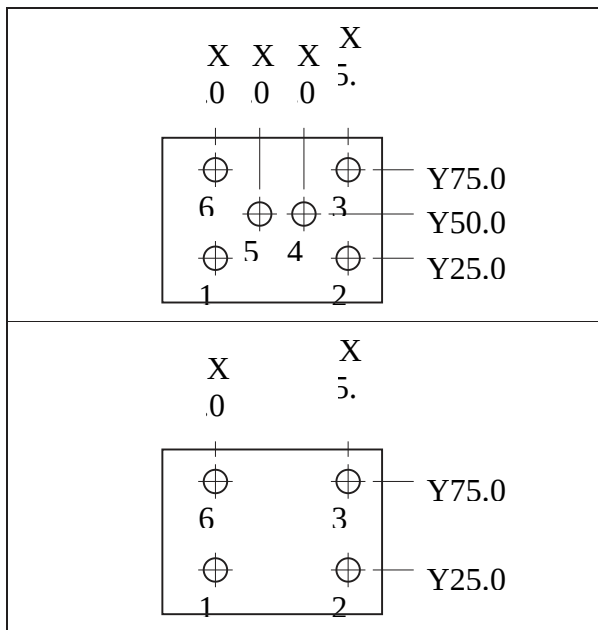
/ N20 G00 X43.0

N21 X400.0 Z45.0 T0600 M01

Program O2303 demonstrates a single program for two parts with similar characteristics. One part requires a single groove, the other requires two grooves on the same diameter. In the example, both grooves are identical - they have the same width and depth and are machined with the same tool. The only difference between the two examples is the number of grooves and the second groove position. Machining the part will require block skip function set ON or OFF, depending on drawing requirements.

Evaluate the more important blocks in the program example. Block N15 is the initial tool motion to the start of the first groove at Z-20.0. In the next two blocks, N16 and N17, the groove will be cut, and the tool returns to the clearance diameter. The following three blocks will cut the second groove, if it is required. That is the reason for block skip function. In block N18, tool moves to the initial position of groove 2 at Z-50.0, in block N19 the groove is cut. In block N20, tool retracts from the groove to a safe clearance position.

Milling example shown in *Figure 23-4*, also in metric, is represented in program O2304. This program handles two similar patterns that have four identical holes for both parts and two missing holes in the second part only. This is a good example of similar parts program, using block skip.



Both variations of program O2304 machine a hole pattern with either 6 or 4 holes. Block skip function has been used to make a single program covering *both* patterns. Top of *Figure 23-4* shows the hole pattern when block skip function is set OFF, the bottom shows the hole pattern when block skip mode is set ON.

O2304 (MILLING EXAMPLE)

N1 G21

...

N16 G90 G00 G54 X30.0 Y25.0 M08

N17 G43 Z25.0 S1200 M03 H04

N18 G99 G81 R2.5 Z-4.0 F100.0 (HOLE

N19 X105.0 (HOLE

N20 Y75.0 (HOLE

/ N21 X80.0 Y50.0 (HOLE

/ N22 X55.0 (HOLE

N23 G98 X30.0 Y75.0 (HOLE

**N24 G80 G28 X30.0 Y75.0 Z25.0
5 M01**

Blocks N18 to N20 will drill holes 1, 2 and 3. Hole 4 in N21 and hole 5 in N22 will be drilled only if the block skip function is set to inactive mode (OFF), but neither one will be drilled when block skip setting is active (ON). Block N23 will always drill hole number 6.

A variation of this application is program O2305. There are *five-hole* positions, but the block skip function is used *within* a block, to control only the Y-position of the hole. Top of Figure 23-5 shows the hole pattern when block skip function is set OFF, the bottom shows the hole pattern when block skip mode is set ON. The middle hole will have a different Y-axis position, depending on the setting of the block skip function during machining.

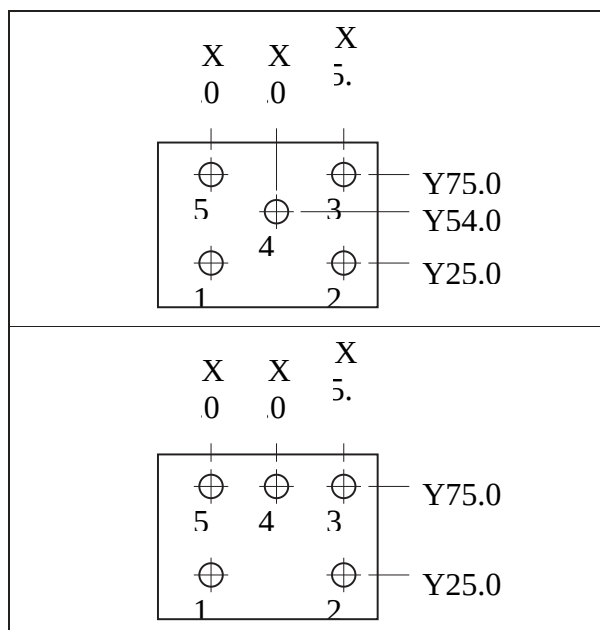


Figure 23-4

Program O2304 - variable machining pattern for a milling application result with block skip OFF (top) and ON (bottom)

Figure 23-5

Program O2305 - variable machining pattern for a milling application result with block skip OFF (top) and ON (bottom)

O2305 (MILLING EXAMPLE)

N1 G21

...

**N16 G90 G00 G54 X30.0 Y25.0
J8**

N17 G43 Z25.0 S1200 M03 H04

N18 G99 G81 R2.5 Z-4.0 F100.0 (HOLE 1)

N19 X105.0 (HOLE 2)

N20 Y75.0 (HOLE 3)

**N21 X67.0 / Y54.0 (==>
HOLE 4)**

N22 G98 X30.0 Y75.0 (HOLE 5)

N23 G80 G28 X30.0 Y75.0 Z25.0 N24 M01

Hole 4 in block N21 will be drilled at the location of X67.0Y75.0, if the block skip mode is ON. The address Y54.0 in block N21 will not be processed. If the block skip mode is OFF, hole 4 will be drilled at coordinate location of X67.0Y54.0. In that case, the Y75.0 position from block N20 will be overridden. In order to guarantee proper drilling at position 5, the coordinate Y75.0 in block N22 must be written. If it is omitted, Y54.0 from block N22 will take precedence in block skip OFF mode.

Using block skip feature is the simplest way of designing a family of similar parts. These applications are a bit limited with block skip function alone, but they offer the fundamentals of a powerful programming technique and an example of logical thinking. Many detailed explanations and examples of programming complex families of parts can be found in a special *Custom Macro* option Fanuc offers on most control systems.

Trial Cut for Measuring

Another useful application of block skip is to provide the machine operator with means of measuring the part *before* any final machining has been done. Due to various dimensional imperfections of a cutting tool combined with other factors, the completed part may be slightly outside of the required tolerance range.

The following method of programming is very useful for programming parts requiring very close tolerances. It is also a useful method for those parts, where the part shape is difficult to measure after all machining is completed, for example conical shapes, such as tapers. The same method is also quite useful for parts where the cycle time of an individual tool is relatively long, and all tool offsets have to be fine-tuned *before* production machining.

This approach to part programming is more efficient, as it eliminates a recut, increases surface finish, and can even prevent a scrap. In either case, *trial cut* programming method that employs the block skip function is used. Setting the block skip mode OFF, machine operator checks the trial dimension, adjusts the suitable offset, if necessary, and continues machining with block skip set ON.

All general concepts described in example O2306 are equally applicable to turning and milling - *Figure 23-6*.

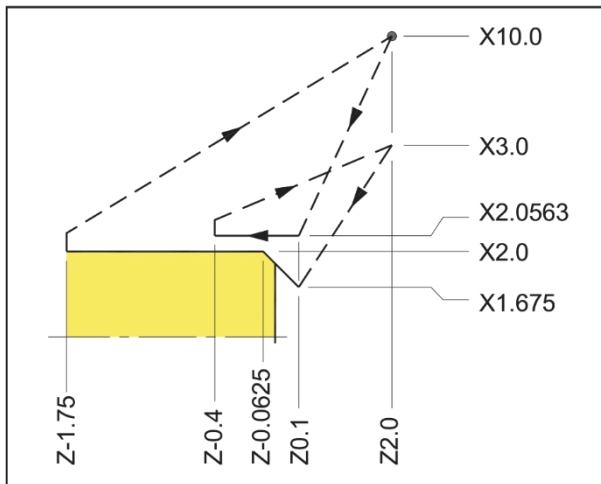


Figure 23-6

Application of a trial cut for measuring on a lathe - program O2306

O2306

(TRIAL CUT - LATHE)

N1 G20

...

N10 G50 S1400

N11 G00 T0600 M43

N12 G96 S600 M03

/ N13 G42 X2.0563 Z0.1 T0606 M08

/ N14 G01 Z-0.4 F0.008

/ N15 X2.3 F0.03

/ N16 G00 G40 X3.0 Z2.0 T0600 M00

/ (TRIAL DIA IS 2.0563 INCHES)

/ N17 G96 S600 M03

N18 G00 G42 X1.675 Z0.1 T0606 M08

N19 G01 X2.0 Z-0.0625 F0.007

N20 Z-1.75

N21 X3.5 F0.01

N22 G00 G40 X10.0 Z2.0 T0600 N23 M01

When program O2306 is processed with block skip set OFF, all blocks will be executed, including the trial cut and finish profile. With block skip set ON, the only operation executed will be finishing to size, *without* the trial cut. In this case, all significant instructions are retained by repetition of the key commands (block N18). Such a repetition is very crucial for successful processing in both modes of the block skip function. M00 function in N16 always stops the machine and enables a dimensional check.

Selecting trial diameter of 2.0563 in the example may be questioned. What is the logic for it?

The trial diameter can be other reasonable size, say 2.05. That would leave 0.025 stock per side for the finish cut. It is true that a different diameter *could* have been selected. The four decimal number was only selected for one reason - *to psychologically* encourage the operator to maintain accurate offset settings. Feel free to disagree - programmers may prefer a three or even two decimal number instead - the options are open.

In the next example, another trial cut will also be programmed before actual machining, but for a much different reason - *Figure 23-7*.

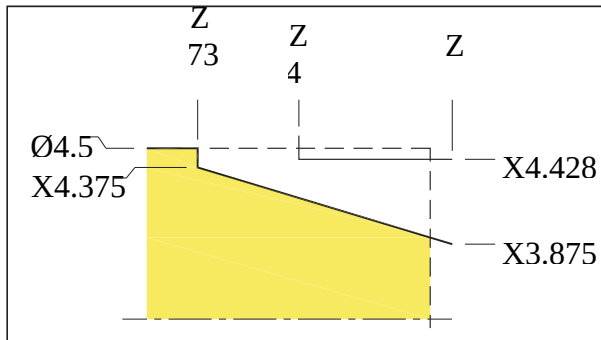


Figure 23-7

Trial cut for taper cutting on a lathe - program O2307

In program O2307, the part finished shape is a taper, a feature difficult to measure when completed. Adjusting the tool offset by trial and error is not the right solution. Programming a trial cut within the area of solid material, *along a straight diameter*, enables the operator to check the trial dimension comfortably and to adjust offset *before* cutting the finished taper.

O2307

(TRIAL CUT FOR TAPER - ONE TOOL)

N1 G20 G99 G40

N2 G50 S1750 T0200 M42

N3 G96 S500 M03

/ N4 G00 G42 X4.428 Z0.1 T0202 M08

/ N5 G01 Z-0.4 F0.008

/ N6 U0.2 F0.03

/ N7 G00 G40 X10.0 Z5.0 T0200 M00

/ (TRIAL CUT DIA IS 4.428 INCHES)

/ N8 G96 S500 M03

N9 G00 G42 X4.6 Z0.1 T0202 M08

N10 G71 U0.15 R0.03

N11 G71 P12 Q14 U0.06 W0.005 F0.01

N12 G00 X3.875

N13 G01 X4.375 Z-0.73 F0.008

N14 X4.6 F0.012

N15 S550 M43

N16 G70 P12 Q14

N17 G00 G40 X10.0 Z5.0 T0200 M01

Program O2307 illustrates a common situation, where a single cutting tool is used for both roughing *and* finishing operations. It shows a logical way of using the block skip function, in a simple form. In most applications, separate tools for roughing and finishing may be needed, depending on the degree of required accuracy. When using two cutting tools, the trial cut dimension is usually more important for the *finishing* tool than for the roughing tool. In program O2308, the block skip function is illustrated using two cutting tools - T02 is for roughing, T04 is for finishing. Previous example in *Figure 23-7* is used.

O2308

(TRIAL CUT FOR TAPER - TWO TOOLS)

N1 G20 G99 G40

N2 G50 S1750 T0200 M42

N3 G96 S500 M03

/ N4 G00 G42 X4.46 Z0.1 T0202 M08

/ N5 G01 Z-0.4 F0.008

/ N6 U0.2 F0.03

/ N7 G00 G40 X10.0 Z5.0 T0200 M00

/ (T02 TRIAL CUT DIA IS 4.46 INCHES)

/ N8 G50 S1750 T0400 M43

/ N9 G96 S550 M03

/ N10 G00 G42 X4.428 Z0.1 T0404 M08

/ N11 G01 Z-0.4 F0.008

/ N12 U0.2 F0.03

/ N13 G00 G40 X10.0 Z5.0 T0400 M00

/ (T04 TRIAL CUT DIA IS 4.428 INCHES)

/ N14 G50 S1750 T0200 M42

/ N15 G96 S500 M03

N16 G00 G42 X4.6 Z0.1 T0202 M08

N17 G71 U0.15 R0.03

N18 G71 P19 Q21 U0.06 W0.005 F0.01

N19 G00 X3.875

N20 G01 X4.375 Z-0.73 F0.008

N21 X4.6 F0.012

N22 G00 G40 X10.0 Z5.0 T0200 M01

N23 G50 S1750 T0400 M43

N24 G96 S550 M03

N25 G00 G42 X122.0 Z3.0 T0404 M08

N26 G70 P19 Q21

N27 G00 G40 X10.0 Z5.0 T0400 M09

N28 M30

%

Example O2308 can be improved further by including the control of taper on the width, for example. Programming a trial cut is useful but often a neglected technique, although it does present many possible applications.

Program Proving

Block skip function can also be useful to prove a new program on the machine, to check it against obvious errors. Operators with limited experience may be uneasy to run a program for the first time. One of their most common concerns is the initial rapid motion towards a part, particularly when clearances are small. Rapid motion rate of many modern CNC machines can be very high, well over 2000 in/min. At such high speeds, rapid approach to the cutting position on the part may not add to operator's confidence, particularly when the approach is programmed too close to material. On most controls, CNC operator can set the rapid override rate to 100%, 50%, 25% and slower. On older controls, the rapid rate override cannot be done.

The next two examples, O2309 and O2310, show a typical programming method to eliminate the problem during setup and program proving, yet maintain full rapid motion rate during repeated operations for productivity.

Block skip function in these examples takes a less usual role - it is used for a *section* of a block, rather than the whole block itself, *if the control supports such a method*.

O2309 (TURNING EXAMPLE)

N1 G20 G40 G99

N2 G50 S2000

N3 G00 T0200 M42

N4 G96 S400 M03

N5 G41 X2.75 Z0 T0202 M08 / G01 F0.1

N6 G01 X.. F0.004 N7 ...

O2310 (MILLING EXAMPLE)

N1 G20 G17 G40 G80

N2 G90 G00 G54 X219.0 Y75.0 M08

N3 G43 Z-1.0 S600 M03 H01 / G01 F30.0

N4 G01 X.. F12.0 N5 ...

Both examples show block skip used *in* a single block. Design of both programs takes advantage of *two conflicting commands within the same block*. If two conflicting commands exist in a single block, the *latter* command used in the block will become effective (exceptions exist).

In both examples, the *first* command is G00, the *second* one is G01. Normally, the G01 motion will take a priority. Because of the slash function, the control will accept G00, if block skip is set ON, but it will accept G01, if block skip is set OFF. When the block skip mode is OFF, both motion commands will be processed and the *second* command in that block becomes effective (G01 overrides G00). Watch for one drawback, already emphasized:

Block skip *within* a block may not work with all controls

During the first machine run, operator should set block skip OFF, making G01 command effective. The tool motion will be slower than in rapid mode, but much safer. Also, feed rate override switch of the control system will become effective, offering additional flexibility.

When program proving is completed and the safe tool approach is confirmed, block skip can be set ON, to prevent the G01 motion from being processed. Both O2309 and O2310 are typical examples of breaking with tradition to achieve a specific result.

Bar feeder Application

On a CNC lathe, block skip function can be used in bar feeding, for continuous machining. If the bar feeder allows it, the techniques is quite simple. Typical program will have *two ends* - one will use the M99 function, the other end will use the M30 function. M99 block will be preceded by the block skip function [/] and will be placed *before* the M30 code in the part program. This special technique is detailed on *page 430*.

Numbered Block Skip

For most machining, block skip function is set to either the ON or OFF position and remains in this mode for the *whole* program. If the ON setting is required for one section of the program, but not for another, the operator has to be informed, usually in program comments. This practice of changing block skip mode in the middle of a program can be unsafe and possibly create major problems.

An optional feature on some controls is a *selective* or a *numbered* block skip function. This option allows the operator to select which portions of the program require the ON setting and which portions require the OFF setting. Setting can be done *before* pressing the *Cycle Start* key to initialize program processing. This method also uses the slash symbol, but followed by an integer, within the range of 1 to 9. Actual selection of the mode is done on the control screen (*Settings*), under the matching skip number.

For example, a program may contain three groups, each expecting a different setting of the skip function. By using the switch number after the slash symbol, all groups are clearly defined, and all the operators must do is to *match* the control settings with the required activity.

N1 ... N2 ..	
/1 N3 ...	(BLOCK SKIP GROUP 1)
/1 N4 ...	(BLOCK SKIP GROUP 1)
...	
...	
N16 ...	
/2 N17 ...	(BLOCK SKIP GROUP 2)
/2 N18 ...	(BLOCK SKIP GROUP 2)
/2 N19 ...	(BLOCK SKIP GROUP 2)
...	
...	
N29 ...	
/3 N30 ...	(BLOCK SKIP GROUP 3)
/3 N31 ...	(BLOCK SKIP GROUP 3)
... ..	
N45 ...	

Identical rules apply for selective block skip function as for the normal version. Incidentally, the /1 selection is the same as a plain slash only, so blocks N3 and N4 above, could have also been written this way:

/ **N3** ...

/ **N4** ...

Numbered block skip function is not available on all controls

Programs using selective block skip function can be very clever and even efficient, but they may place quite a burden on the machine operator. For the majority of jobs, there will be a plenty of programming power available by using the standard block skip function.

22. DWELL COMMAND

Dwell is a technical name for a *pause* in CNC programs it is an intentional time delay applied during program processing, at the machine. In this time period - specified in the program - any axis motion is stopped, while all other program commands and functions remain unchanged (function normally). When the specified time expires, control system resumes processing of program block immediately *following* the dwell command.

A. PROGRAMMING APPLICATIONS

Programming a dwell is very easy and can be quite useful in two main applications:

During actual cutting, when the tool is in contact with material

For operation of machine accessories, when no cutting takes place

Each application is equally important to programmers, although the two are not used simultaneously.

Applications for Cutting

When cutting tool is removing material stock, it is in contact with the machined part. A dwell can be applied during machining for a number of reasons. If the spindle is running, the spindle speed (r/min) is very important.

In practice, the application of dwell during a cut may be used for breaking chips while drilling, grooving, countersinking, counterboring, or parting-off. Dwell may also be used while turning or boring, in order to eliminate any physical marks left on the part by end thrust of the cutting tool. This thrust is the result attributed to various tool pressures during cutting. In many other applications, the dwell function is useful to control cutting feed deceleration on a corner during very fast feed rates, for example. This use of dwell could be particularly useful for older control systems with a possible backlash problem. In both cases, specified dwell command ‘forces’ the machining operation to be *fully completed* in one block, *before* the next block can be executed. CNC programmer still has to supply the exact length of time required for the pause duration. This time has to be sufficient - neither too short nor too long.

Dwell command is always completed before the next operation begins

Applications for Accessories

The second common application of dwell is after certain miscellaneous functions - M functions. Several such functions are used to control a variety of CNC machine accessories, such as bar feeder, tailstock, quill, part catcher, custom features, and many others. Programmed dwell time must allow for the full completion of a certain procedure, such as the operation of a tailstock. During such a procedure, the machine spindle may be either stationary or rotating. Since there will be no contact of the cutting tool with part material in this situation, it is *not* important whether the machine spindle actually rotates or not.

On some CNC machines, the dwell command may also be needed when changing spindle speed, usually after gear range change. This applies mainly to CNC lathes. In these cases, the best advice as to *how* and *when* to program dwell time is to follow the CNC machine manufacturer's recommendation. Typical examples of a dwell used for lathe accessories are described in *Chapter 44*, covering the subject.

B. DWELL COMMAND

G04	Dwell command

Standard preparatory command for dwell is G04. Like other G commands, G04 used by itself

only will do nothing. It must always be used with another address, in this case, specifying the *length of time* to dwell (pause). Three addresses can be used for dwell - they are X, P or U (address U can only be used for CNC lathes). Actual time duration specified by the selected address is either in *milliseconds*, or in *seconds*, depending on which address was used. Some control systems may use a different address for programming dwell, but its main purpose as well as programming methods remain identical.

Several fixed cycles for CNC machining centers also use dwell. This dwell is programmed together with the cycle data, *not* in a separate block (G04 is not used). Only fixed cycles that require a dwell time can use it in the same block. For all other applications, the dwell command must be programmed as an *independent block*. Dwell will always remain active for one block only and does not carry over to the next block. Dwell is a one-block function - it is *not* modal. During dwell execution, there is no change in status of program processing, only the overall cycle time will be affected.

Dwell Command Structure

The structure - or format - for the dwell time is:

X5.3 ... *All machines, excluding fixed cycles*

U5.3 ... *Lathes only*

P53 ... *All machines, including fixed cycles*

In any case, the typical representation is five digits *before* and three digits *after* the decimal point, although that may vary on different control systems.

Since milliseconds or even seconds can be used as units of dwell, their relationship can be established:

1s 1000ms
1ms 0 001. s

+ *where* ...

s = second

ms = millisecond

Examples of practical application of the dwell format are:

G04 X2.0 ...*preferred for long dwells in seconds*

G04 P2000 ...*preferred for short or medium dwells in ms*

G04 U2.0 ...*lathe only in seconds*

In these examples, the dwell is 2 seconds or 2000 milliseconds. All three formats are shown with identical results. The next example is similar:

G04 X0.5

G04 P500

G04 U0.5

This example illustrates a dwell of 500 milliseconds, or one half of a second. Again, all three formats are shown.

In a CNC program, the dwell function may be used in the following way - note the dwell is programmed in a separate block, between two motions:

N21 G01 Z-1.5 F12.0

N22 G04 X0.3 (DWELL COMMAND 0.3 SEC)

N23 Z-2.7 F8.0

Programs using X or U addresses may cause a possible confusion, particularly to new programmers. The X and U addresses may incorrectly be interpreted as an axis motion. *This will never be the case.* By definition, the X axis and its lathe application, the U axis, is the *dwelling* axis.

X axis is the only axis common to all CNC machines

No axis motion will take place when the X, P or U address is used with the dwell command G04

The control unit interprets commands X or U as dwell, *not as axis motion*. This is because of presence of the preparatory command G04, which establishes *meaning* of the address that follows it. If using the X or U address for dwell does not feel comfortable, use a third alternative - the address P. Keep in mind, that the P-address does *not* accept decimal point, so any dwell is programmed directly as the *number of milliseconds* in order to control the elapsed time. One millisecond is 1/1000th of a second, therefore one second is equivalent to 1000 milliseconds.

Both addresses X and U can also be programmed in milliseconds with G04, without a decimal point - for example,

G04 X2.0 is equal to **G04 X2000**

Leading zero suppression is assumed in the format without decimal point (trailing zeros are always required):

P1 = **P0001** ... *1millisecond* = *0.001second*

P10 = **P0010** ... *10milliseconds* = *0.01second*

P100 = **P0100** ... *100milliseconds* = *0.1second*

P1000 = **P1000** ... *1000milliseconds* = *1.0second*

While minimum dwell is more important than maximum dwell, each control has a limit for maximum dwell duration. For the format using five digits in front of a decimal point and three digits following it, the dwell range is between 0.001 and 99999.999 seconds. That is equivalent to a dwell range from the minimum of 1/1000th of a second, up to *27 hours, 46 minutes and 39.999 seconds*.

Dwell programming applications using the X and P addresses are identical to both machining centers and lathes, but the U address can only be used in lathe programs. The selection of either metric or imperial dimensional units has no effect on the dwell function whatsoever, as time is not dimensional.

C. DWELL TIME SELECTION

Seldom ever the dwell time will exceed more than just a few seconds, most often much less than one second. Dwell is always a *nonproductive time*, and it should be selected as the shortest time needed to accomplish the required action. Any time delay for completion of a particular machine operation or a special machine accessory is usually recommended by the machine manufacturer. Selecting dwell time for cutting purposes is always programmer's responsibility. Unfortunately, some programmers often specify the dwell time too high. After all, one second seems like a very short time, but think about a typical example:

In one program block, a dwell function is assigned for the duration of one second. The spindle speed is set to 480 r/min and the dwell is applied at 50-part locations, perhaps during a spot face operation. That means the cycle time for *each* part is 50 seconds longer *with* the dwell, then it would be *without* the dwell. Fifty seconds may not seem too unreasonable, but are these precious

seconds necessary? Give it a little thought or - even better - calculate it. If dwell must be used at all, make sure to calculate the *minimum* dwell that can do the job. It is easy to select dwell arbitrarily, by guessing and without much thinking. In the example, the minimum dwell required is only 0.125 seconds (!):

$$60 / 480 = 0.125$$

This minimum dwell is *eight times shorter* than the one second dwell programmed initially. If minimum dwell is programmed rather than the estimated dwell, the cycle time will increase by only 6.25 seconds, rather than the original 50 seconds - a significant improvement in programming efficiency and productivity improvement at the machine.

Minimum dwell calculation and other issues related to it are described shortly.

D. SETTING MODE AND DWELL

Most programs for machining centers will use feed rate per time (programmed in millimeters per minute - *mm/min* - or inches per minute - *in/min*). Lathe applications are normally programmed as feed rate per one spindle revolution, in millimeters per revolution - *mm/rev* - or inches per revolution - *in/rev*. On many Fanuc controls, a parameter setting allows programming a dwell in either as elapsed time in seconds or milliseconds - *or the number of spindle revolutions*. Each application has its practical uses and benefits. Depending on system parameter setting, dwell command will assume an incompatible meaning with each setting:

Time Setting

This is the most common and practical default setting for virtually all CNC machines. For the *elapsed time* setting, the dwell is always programmed in seconds or milliseconds, within the range allowed by the control unit, for example, from 0.001 to 99999.999 seconds, a typical range for Fanuc and many similar control systems:

G04 P1000

... represents the dwell of one second, equivalent to 1000 milliseconds.

Number of Revolutions Setting

For the number of *spindle revolutions* setting, the dwell is expressed as the number of times a spindle rotates, within the range of 0.001 to 99999.999 *revolutions*, for example:

G04 P1000

... represents the dwell duration of one *revolution* of the spindle (P1000 = 1.000).

E. MINIMUM DWELL

During a cut, that is for operations where cutting tool is in contact with the machined part, minimum dwell definition is important, but the setting mode is unimportant (time or number of revolutions).

Minimum dwell is the time required to complete one revolution of the spindle

Minimum dwell, programmed in seconds, can be calculated, using a simple formula:

Minimum dwell (sec)	$\frac{60}{n} \text{ ——— } r /$

Example:

To calculate minimum dwell in seconds for spindle rotation of 420 r/min, divide the *r/min* into sixty (there are 60 seconds in one minute):

$$60 / 420 = 0.143 \text{ second's dwell}$$

Format selection of dwell block in the program will vary, depending on the machine type used and a particular programming style. All following examples represent the same dwell time of 0.143 of a second:

G04 X0.143

G04 P143

G04 U0.143

Regardless which format is used, all dwell times in the example specify 143 milliseconds, which is 0.143 of a second. Dwell in *milliseconds* can be calculated directly:

	60	1000	60000	_____
Minimum dwell ms ()				r / min r / min

It is allowed to mix both formats in a program, but such a practice does not represent consistent programming style.

Practical Considerations

For *practical* dwell applications, the calculated minimum dwell is only *theoretically* correct and may *not* be the most practical to use. It is always better to round off the calculated value of minimum dwell *upwards* to get *more* than one spindle revolution. For example,

G04 X0.143 may become **G04 X0.2**

If double time is used - then G04 X0.143 will become G04 X0.286, or even G04 X0.3 to round off the time.

The reasoning for such an adjustment takes into consideration machining realities. It is common that machine operator may be running a certain job with spindle speed in an override mode, perhaps even set at its lowest setting, at 50%. Since 50% spindle speed override is the usual minimum setting on most CNC controls, the *double* minimum dwell will *always guarantee at least one complete revolution*, without any loss of production time.

Is there a magic dwell that can be used for most jobs? Not really, but you will find that most dwells will be very short, usually within the range of 0.2 and 0.5 seconds. A reversal of the previous formula calculates r/min for a given dwell:

60
r / min _____
Dwell _{sec}

For example, if programmed dwell is 0.25 of a second is suitable for rather slow speed of 240 r/min. In fact, the higher the spindle speed, the shorter dwell is required.

Another example is based on a particular machining *operation*, for example, a spot drill. Let's say, a spot drill is usually programmed within the spindle speed range of 600 to 1500 r/min, depending on stock material. The double minimum dwell for 600 r/min is 200 ms, while for 1500 r/min it is only 80 ms. The practical benefit is that if you program 250 ms for *any* spindle speed below 1500 r/min, the double minimum dwell will always be guaranteed. It is a compromise, but for a few holes it may be the fastest way to program dwell for spot drilling or similar operations.

F. NUMBER OF REVOLUTIONS

In the other dwell mode (selected by a system parameter), the programming format will only *appear* to be the same, but its meaning will be much different. In some applications, it may be desirable to program a delay for a certain *number of spindle revolutions*, rather than *elapsed time*.

In lathe grooving, when the tool reaches groove bottom, dwell time may be required to clean up the groove bottom diameter. Calculating the time in seconds (see below) would solve the problem, but an alternative approach could be more attractive. Many controls offer programming the required number of spindle revolutions directly, providing a system parameter is set. For example, to dwell as long as it takes to complete *three spindle revolutions* can be programmed directly, regardless of the spindle r/min.

It is normal in CNC lathe programming to use *cutting speed* rather than *spindle speed* in grooving. This situation presents a small problem - the actual spindle speed is *not* known. Yes, it can be calculated, but the effort may not be worth it. Using number of revolutions will solve the problem, *even if the cutting speed is changed (optimized)*.

System Setting

If the control system is set to accept dwell as the *number of spindle revolutions*, rather than as elapsed time in seconds or milliseconds, the programming is very direct. All that is needed is to call the dwell command G04, followed by the number of required revolutions:

G04 X3.0 ...three spindle revolutions - direct input

G04 P3000 ...three spindle revolutions – direct input

G04 U3.0 ...three spindle revolutions – direct input

Each format represents the same result - a dwell lasting three spindle revolutions. Can we tell from the program whether the value means *time* or *revolutions*? We cannot.

Control settings have to be known for dwell

We *have to know* the control settings. One clue may be somewhat large dwell input - three revolutions are usually much shorter than three seconds of dwell. Note that the decimal accuracy is still written or implied, to allow fractions of a revolution, such as one half or one quarter.

Time Equivalent

The two dwell modes cannot be mixed in one program, and even between programs, such a mix is not practical. CNC machine system parameter can be set to only one dwell mode at a time. Since control parameters are normally set for dwell as elapsed time, rather than spindle revolutions, the equivalent time must be calculated. The spindle speed (r/min) must always be known in such case.

Equivalent dwell time in seconds can be calculated to be equal to the required number of spindle revolutions.

Use the following formula:

Dwell _{sec}	$\frac{60 \text{ } n}{\text{r / min}}$

+ where . . .

60 = Number of minutes (conversion factor) n = Required number of spindle revolutions
 revolutions r/min = Current spindle speed (revolutions per minute)

Example :

To calculate dwell time in seconds for *full three* spindle revolutions, at spindle speed of 420 r/min, the above formula can be applied:

Dwell_{sec} = 60 3 / 420 = 0.429

Program block representing the required three spindle revolutions in terms of dwell *time* will take one of the following forms:

G04 X0.429 ...three spindle revolutions – time equivalent

G04 P429 ...three spindle revolutions – time equivalent

G04 U0.429 ...three spindle revolutions -time equivalent

It may also be a good idea to work backwards and calculate the *equivalent of dwell time*, represented as the number of spindle revolutions. Usually, the result will not be an integer number and will require rounding to the nearest value upwards. The above formula can be easily modified to:

	r / min	Dwell
Dwell _{rev}		_____sec
	60	

Example:

To confirm that the formula is correct, use the value of 0.429 from the previous example and calculate the number of revolutions for a dwell of 0.429 seconds at 420 r/min:

$$\text{Dwell}_{\text{rev}} = 420 \cdot 0.429 / 60 = 3.003 \text{ revolutions}$$

Result confirms the formula is correct. It is quite likely that the calculation will start with a dwell that is already rounded, for example, to one half of a second:

$$\text{Dwell}_{\text{rev}} = 420 \cdot 0.5 / 60 = 3.5 \text{ revolutions}$$

The dwell time based on a required number of revolutions is mostly used for CNC lathe applications, especially when cutting with very slow spindle speeds. A slow spindle rotation does not have the latitude of faster speeds and does not allow for a large error in dwell calculation. Keep in mind that the goal is to get *at least one complete* part rotation in order to achieve desired machining results. Otherwise, why program dwell at all? Consider another example:

Dwell is programmed for *one half of a second* duration, with spindle rotation set to low 80 r/min. The number of revolutions for one half of a second will be:

$$80 \cdot 0.5 / 60 = 0.6666667$$

which is *less* than one complete spindle revolution. The reason for programming the dwell function in the first place is not honored here and the dwell time has to be increased. Dwell of 0.5 seconds is therefore not sufficient. The minimum dwell has to be *calculated*, using the formula presented earlier:

$$60 \cdot 1 / 80 = 0.75 \text{ seconds}$$

Generally, there is not much use for this type of calculations - most programming assignments can be handled very well with the standard *dwell per time* calculations.

G. LONG DWELL TIME

For general CNC machining purposes, an unusually long dwell time is neither required nor it is necessary. Does that mean long dwell times are not needed at all?

A long dwell time is elapsed time that is well *above the established average* for most normal applications. Seldom ever there is a need to program dwell time during a part machining in excess of one, two, three, or four seconds. The large range available on the control system (*over 27 hours*) is more important to the *maintenance personnel*, than to CNC programmers. As an example of typical application when an unusually long dwell *may* be beneficial, is a program developed by the maintenance technicians for testing the spindle functionality.

Consider carefully the following actual situation common to machine service - a spindle of a CNC machine has been repaired and must be tested before the machine can be released back to production. Testing will include running the spindle at various speeds, for a certain period of time

of each speed selection.

In a typical example, the maintenance department requires a small CNC program, in which machine spindle will rotate for 10 minutes at 100 r/min, then for another 20 minutes at 500 r/min, followed by the spindle rotation at the highest rate of 1500 r/min for additional 30 minutes. Program development is not an absolute necessity, since the maintenance technician may do such a test by manual methods. The manual approach will not be very efficient, but it will still serve the purpose of the maintenance test.

A better choice in these cases is to *store* the testing procedure as a small program, directly into CNC memory. The maintenance (service) program will be a little different for machining centers than for lathes, but all main objectives will remain identical.

Example - Machining Centers - Spindle test:

S100 M03	(100 R/MIN INITIAL SPEED)
G04 X600.0	(600 SECONDS IS 10 MINUTES)
S500	(SPEED INCREASED TO 500 R/MIN)
G04 X1200.0	(1200 SECONDS IS 20 MINUTES)
S1500	(SPEED INCREASED TO 1500 R/MIN)
G04 X1800.0	(1800 SECONDS IS 30 MINUTES)
M05	(SPINDLE STOP)

For machining centers, the program starts with 100 r/min spindle speed. This selection is followed by a 600 second dwell (10 minutes). Spindle speed is then increased to 500 r/min and dwell time to 1200 seconds (another 20 minutes). The last speed is 1500 r/min running for 1800 seconds, or another 30 minutes, before the spindle finally stops.

Note that in cases of especially long dwell, using *seconds* rather than *milliseconds* is more practical.

Example - Lathes - Spindle test:

M43	(GEAR RANGE SELECTION)
G97 S100 M03	(100 R/MIN INITIAL SPEED)
G04 X600.0	(600 SECONDS IS 10 MINUTES)
S500	(SPEED INCREASED TO 500 R/MIN)
G04 X1200.0	(1200 SECONDS IS 20 MINUTES)
S1500	(SPEED INCREASED TO 1500 R/MIN)
G04 X1800.0	(1800 SECONDS IS 30 MINUTES)
M05	(SPINDLE STOP)

This lathe example is similar to the previous one. Initial spindle speed setting includes gear range selection, for example, M43. Spindle rotation has been set to 100 r/min. Dwell of 600 seconds follows, leaving the spindle rotating for full 10 minutes. Then the speed is increased to 500 r/min and remains that way for another 20 minutes (1200 seconds). Before the spindle is stopped, one more change is done - the spindle speed increases to 1500 r/min and remains at that speed for another 30 minutes (1800 seconds).

Long dwells are for special applications only and should never be used for other purposes.

Maintain all safety rules when using long dwell times!

Machine Warm-Up

Another program (typically a subprogram) that uses a long dwell time is favored by many CNC programmers and CNC machine operators, to 'warm-up' the machine before running a critical job. This machine warming activity takes place typically at the start of a morning shift during winter months or in a cold shop. This programming approach allows CNC machine to reach a certain ambient temperature before any precision components are machined. The same approach can also be used to gradually reach the maximum spindle speed for high-speed machining (8000 r/min and up). As always, all safety considerations must have a high priority in these cases.

X Axis is the Dwelling Axis

The control display screen shows how much time is still left before the dwell time expires. This can be viewed by looking at the X display of the *Distance-To-Go* indicator on the POS (position) screen of a typical Fanuc control. This display will always be displayed as X, since X axis is the only dwelling axis, regardless of control system, even if addresses P or U are programmed. Why the X axis has been selected as the dwelling axis and not any other axis? There is a simple reason - because *the X axis is the only axis common to all CNC machine tools* - i.e., drilling machines, mills, machining centers, flame cutters, waterjets, lasers, and so on. They all use XY or XYZ axes. Two-axis lathes use XZ axes (there is no Y axis) and wire EDM uses XY axes (there is no Z axis). Other machines are similar.

Safety and Dwell

A few safety reminders have been mentioned already always exercise caution when using long dwells, particularly for service or maintenance purposes. No CNC machine should *ever* be left completely unattended - after all, a test can fail. For long testing times, proper warning signs should be prominently posted at the machine. If signs are not practical, personal supervision will be necessary.

One more note - the dwell function should never be programmed for the purpose of allowing the machine operator time to perform short manual operation while program is running. Manual jobs such as polishing, filing, deburring, part reversal, tool or insert change, chip removal, inspection, lubrication, etc., must always be done - if absolutely necessary during program execution - as a manual operation, *never under the program control!*

Never use dwell to perform manual operations on the machine!

H. FIXED CYCLES AND DWELL

Chapter 25 covers fixed cycles for CNC machining centers in significant detail. Descriptions of all cycles can be found there. For this chapter, here are some comments as how dwell is programmed with fixed cycles. Several fixed cycles can be programmed with a dwell:

Normally, cycles G76, G82, G88, G89

Also cycles G74 and G84, only by parameter setting

Dwell address in fixed cycles is always P, to avoid duplication of X address in the same block. Neither address U nor command G04 are *ever* programmed in fixed cycles dwell function is 'built' into all cycles that allow it (technically all cycles do). Rules for calculating dwell time are the same for cycles, as for any other machining application.

Example (spot drilling):

N9 G82 X30.0 Y16.0 R2.0 Z-3.75 P300 F150.0

In the example, dwell of 300 milliseconds - 0.3 seconds is specified by the address P. Dwell will become effective upon completion of motion along the Z axis (actual cutting motion), but *before* the rapid return motion.

If a G04 P.. is programmed as a separate block in fixed cycle mode, for example, between G82

block and G80 block, *no cycle* will be executed in that block and the value of P in the fixed cycle definition is *not* updated. On the latest Fanuc controls, a system parameter setting enables or disables this usage. If this rare method is used at all, the command G04 P.. will be active *before* the tool rapid motion from the location just completed. The dwell function will always be executed while the cutting tool is out of a hole, in the clear space. *This feature is seldom required.*

23. MACHINING HOLES

Single point hole operations on a CNC lathe are much more limited than those on a CNC machining center. First, the number of holes that can be drilled or tapped in a single operation on a lathe is only one per part operation (two are rare), while the number of holes for a milling application may be in tens, hundreds and even thousands. Second, the boring (internal turning) on a lathe is a *contouring* operation, unlike boring on a milling machine, which is a *point to-point* operation.

All point-to-point machining operations on a CNC lathe are limited to those that can be machined with a cutting tool positioned at the spindle *centerline*. Typically, these operations include center drilling, standard drilling, reaming and tapping. A variety of other cutting tools may also be used, for example, a center cutting end mill (slot drill) to open up a hole or to make a flat bottom hole. An internal burnishing tool may also be used for operations such as precise sizing of a hole, etc. To a lesser degree, other operations, such as counterboring and countersinking may be programmed at the lathe spindle centerline, with a special point-to-point tool - *not a contouring tool*. All operations in this group will have one common denominator - they are all used at the spindle centerline and programmed with the X position as X0 in the program block.

Spindle speed for *all* centerline operations on a CNC lathe must be programmed in actual revolutions per minute (*r/min*), *not* in the constant surface speed mode (CSS), also known as CS - cutting speed. For that reason, G97 is used for example,

G97 S575 M03

will assure the required 575 r/min at the normal spindle rotation (at 100% spindle speed override).

What will happen if CSS mode is used with G96 command, rather than the proper G97 command? CNC system will use the given information, spindle speed address S, in the program (given in peripheral - or surface - speed per minute, as *m/min* or *ft/min*). The system will then calculate the required spindle speed in *r/min* for the use by the machine tool.

Such a calculation is based on standard mathematical formula that relates to part *diameter*. If the diameter is zero which is *exactly* what it is at the spindle centerline - the spindle revolutions will always be the *highest r/min* that is available within currently selected spindle gear range. This calculation is an exception to standard r/min calculation formula, where spindle speed at the centerline (diameter zero) would be zero - yes, *0 r/min*!

For example, if the cutting (surface) speed for a given material is 450 ft/min, the r/min at 3.0 inches (X3.0) for the same material will be approximately:

$$S = (450 \cdot 3.82) / 3 = 573 \text{ rpm}$$

If the same speed of 450 ft/min is applied to the diameter zero (X0 in the program), the formula does not change, but the resulting action does:

$$S = (450 \cdot 3.82) / 0 = \text{results in error}$$

Although the spindle might be expected to stop control issue an error condition, it will do the *exact opposite* (because of the control design). Spindle speed will reach the maximum *r/min* that the current gear range will allow. Be very careful here - make sure that all centerline operations on a CNC lathe are always done in G97 (r/min) mode and *not* in G96 mode (CSS) mode.

A. TOOL APPROACH MOTION

A typical geometry offset configuration setup (or the old G50 values) on a CNC lathe often have a relatively large X value and relatively small Z value. For example, metric geometry offset for a tool may be X-300.0 Z-25.0 (or programmed as G50 X300.0 Z25.0). This location indicates

a suitable tool change position applicable to a drill. What does it mean to the tool motion for a drilling operation?

It means that machine rapid motion will complete the Z-axis motion long before completing the X-axis motion (with the quite common hockey-stick motion of the rapid command). Resulting tool motion is located *very close* to the part face (G21 mode):

N36 T0200 M42

N37 G97 S700 M03 N38 G00 X0 Z2.5 T0202 M08 N39 ...

To avoid a potential collision during tool approach towards the part, use one of the following methods:

Move the X-axis first to the spindle centerline, then the Z-axis, directly to the start position for drilling

Move the Z-axis first to a clear position, then the X-axis to spindle centerline, then complete the Z-axis motion into the start position for drilling

The first method may be practical only in those cases when the tool motion area is absolutely clear and has no obstacles in the way (do not count on such a situation). The second method, and probably the most common in programming, will first move the Z-axis close (but not too close) to the part, say 15 mm in the front (Z15.0). Motion that follows is X-axis motion only - directly to the centerline (X0). At this point, the cutting tool (such as a drill) is far from the Z-axis face. The last approach motion will be to the Z-axis start position, closer to part face, where the actual drilling cut begins. This method eliminates (or at least minimizes) any possibility of collision with obstacles along the way. The obstacles are - or at least could be - tailstock, parts catcher, steady rest, fixture, face plate, etc. An example of this programming tool path method is the previous example, modified (G21 mode):

N36 T0200 M42

N37 G97 S700 M03

N38 G00 X0 Z15.0 T0202 M08 N39 Z2.5 N40 ...

This programming method splits the tool approach along Z-axis into *two* tool positions - one is a safe clearance for approach, the other one is a safe clearance position for the drill start. There is a minor alternative to this motion - the last Z-axis approach will be at a cutting feed rate G01, rather than at a rapid motion rate G00:

N36 T0200 M42

N37 G97 S700 M03

N38 G00 X0 Z15.0 T0202 M08

N39 G01 Z2.5 F2.0

N40 ... F.. (CUTTING FEED RATE)

For the last approach motion, Z-axis motion has been changed to a linear motion, with a relatively high feed rate of 2 mm/rev), followed by cutting feed rate. Feed rate override can be used for setup, to control the approach rate. During actual production, there will be no significant loss in the cycle time.

B. TOOL RETURN MOTION

The same logical rules of motion in space that apply to tool approach, apply also to tool return motion. Remember that the first motion *from* a hole must always be along the Z-axis (G20 mode):

...

N40 G01 Z-0.8563 F0.007

N41 G00 Z0.1 ...

In block N40, the actual drill cutting motion takes place. When the cut is completed, block N41 is executed. The drill will rapid out of the hole to the same position it started from (Z0.1). It is not necessary to return to the same position, but it makes programming style more consistent. Once the cutting tool is safely out of the hole, it has to return to a tool changing position. There are two methods:

Simultaneous motion of both axes

Single axis at a time

Simultaneous motion of the X and Z axes does not present the same problem as it did on approach - on the contrary. Z-axis will complete the motion first, moving away from the part face. Also, there is no reason to fear a collision during a return motion if the approach motion was successful and programming style was consistent:

...

N70 G01 Z-0.8563 F0.007

N71 G00 Z0.1 N72 X11.0 Z2.0 T0200 M09 ...

If in doubt, or if an obstacle is expected to be in the way of a tool motion, for example a tailstock, program a single axis at a time. In most cases, that will move the positive X axis first, as most obstacles would be to the right of the part:

...

N70 G01 Z-0.8563 F0.007

N71 G00 Z0.1 N72 X11.0 N73 Z2.0 T0200 M09 ...

This programming example illustrates return motion with the X-axis programmed first. The fact that the tool is 0.100 off the front face is irrelevant - after all, the tool started cutting *from* that distance without a problem.

Other, less traditional, methods for tool motion towards and away from the part are also possible. **Drilling and Reaming on Lathes**

Drilling on a CNC lathe is also quite common operation, mainly as means for a hole opening to be used with other tools, such as boring bars. There are three basic kinds of drilling, typical to a CNC lathe machining:

Center drilling and spot drilling

Drilling with a twist drill Indexable insert drilling

Each method follows the same programming techniques as those described in the milling section earlier, except that there are no fixed cycles of the milling type used for the lathe work. Keep in mind that on a CNC lathe, the part is rotating, whereby the cutting tool remains stationary. Also keep in mind that most lathe operations take place in a horizontal orientation, causing fewer concerns about coolant direction and chip removal.

C. PECK DRILLING CYCLE - G74

On Fanuc and compatible controls, there is a multiple repetitive cycle G74 available, that can be used for two different machining operations:

Simple roughing with chip breaking

Peck drilling (deep hole drilling)

In this section, the peck drilling usage of G74 cycle is described. Roughing application of G74 cycle is a contouring operation and is very seldom used.

In peck drilling, just like in any ordinary drilling, select spindle speeds and cutting feed rates first, then determine the hole starting position and finally, its depth position. In addition, establish (or even calculate) the depth of each peck. Lathe cycle G74 is rather limited in what it can do, but it has its uses. Its format for peck-drilling is:

G74 X0 Z.. K..

+ where ...

G74= Peck drilling cycle
ection

X0 = Indicates cutting on
terline

Z = Specifies the end point for
lling

K = Depth of each peck
ways positive)

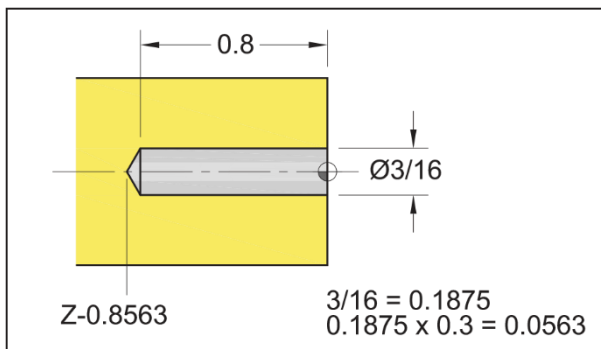


Figure 26-30

Peck drilling example - sample hole

The following program uses illustration in *Figure 26-30*, and shows an example of drilling a 3/16 hole (0.1875) with a peck drill depth of 0.300 inches:

...

N85 T0400 M42

N86 G97 S1200 M03

N87 G00 X0 Z0.2 T0404 M08

N88 G74 X0 Z-0.8563 K0.3 F0.007

N89 G00 X12.0 Z2.0 T0400 M09 N90 M01

Peck drilling motion will start from Z0.2 position in block N87 and continue to Z-0.8563 position in block N88. That results in a 1.0563 long cut. Calculation of the number of pecks is the same as in milling.

With 0.300 length of each peck, there will be total of three *full* length pecks and one partial length peck, at the following Z-axis locations:

Z-0.1

Z-0.4

Z-0.7

Z-0.8563

Although the first three pecks are 0.300 deep each, the first one starts at Z0.2 and ends at Z-0.1. That will result in two thirds of the cut being in the air. Programmer has to decide when this approach is an advantage and when another method would be more suitable. At the end of each peck motion using G74 cycle, the drill will make a small retract by a *fixed* distance. This distance is set by a parameter of the control system and is typically about 0.020 inches (0.5 mm). A full retraction after each peck out of the hole (similar to G83 cycle for milling controls) is not supported by the G74 cycle.

Note that there is *no* programmed motion *out* of the hole when the peck drilling cycle is completed. This return motion is built-in within G74 cycle. If a tool motion such as G00 Z0.2 M05 follows block N88, no harm is done. It may give the operator extra confidence when running the job.

D. TAPPING ON LATHES

Tapping on CNC lathes is a common operation that follows the same machining principles as tapping on machining centers. The major difference for lathes is the absence of a tapping cycle. There is no real need for a tapping cycle on a lathe, since most of lathe tapping operations machine only *one hole of the same type*. The absence of a tapping cycle may present some unexpected difficulties. Unfortunately, they are more common among programmers with limited experience. Before evaluating some of these difficulties closer, it is important to know the actual tool that holds the tap (*tap holder*) and the tapping process on CNC lathes in general.

The selected tap should always be mounted in a special tapping holder; the best type is one with tension and compression features, known as *tap floating holder*. Never use a drill chuck or a similar solid device - it will break the tap quickly and possibly damage the part as well.

Since there is no fixed cycle for tapping on a typical CNC lathe, each tool motion is programmed as a separate block. To do that, and to find out how to tap properly, let's first evaluate the process for a typical right-hand tap in general, applied to a lathe operation:

Step 01	Set coordinate position XZ
Step 02	Select tool and gear range
Step 03	Select spindle speed and rotation
Step 04	Rapid to centerline and clearance with offset
Step 05	Feed-in to the desired depth
Step 06	Stop the spindle
Step 07	Reverse the spindle rotation
Step 08	Feed-out to clear of the part depth
Step 09	Stop the spindle
Step 10	Rapid to the start position
Step 11	Resume normal spindle rotation or end program

Translated into a CNC program carefully, this step-by-step procedure can be used in everyday programming as a general guide to tap on CNC lathes.

Figure 26-31 shows layout of a part and tool setup, used for program example O2607. Program example O2607 follows the eleven steps described above literally and is based on a very solid foundation. Technically, the program O2607 is correct - but only in theory, not practically.

Are there possible problem situations in example O2607 - some thoughts on what can go wrong?

O2607 (TAPPING ON LATHES) (CORRECT VERSION - IN THEORY ONLY) ... (T02 - TAP DRILL 31/64) ...

...

N42 M01

(T03 - 9/16-12 PLUG TAP)

N43 T0300 M42

N44 G97 S450 M03

N45 G00 X0 Z0.5 M08 T0303

N46 G01 Z-0.875 F0.0833

N47 M05

N48 M04

N49 Z0.5

N50 M05

N51 G00 X12.0 Z2.0 T0300 M09

N52 M30

%

A brief look at the program O2607 does not show that anything is wrong. After all, the program covers all necessary motions and is, therefore, correct.

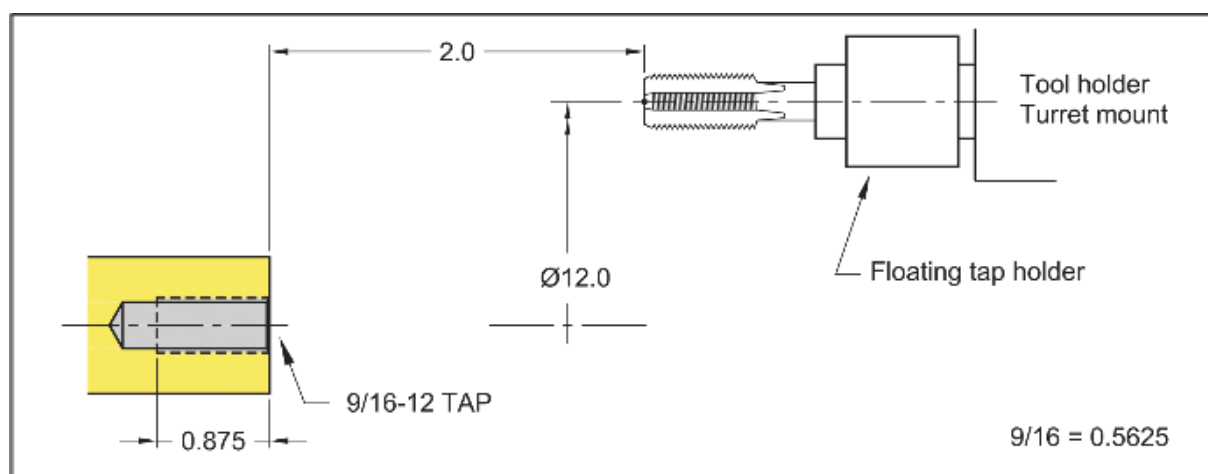


Figure 26-31

Typical setting of a tapping tool on CNC lathe - program examples O2607 and O2608

Yet, this program contains major flaws!

All earlier tapping steps have been carefully followed. Conducting a more in-depth study of the program will reveal two areas of potential difficulty or even danger. The first problem may arise if the machine feed rate override setting switch is *not* set to 100%. Remember, the tapping feed rate is always equal to the thread lead (F0.0833 is the feed rate for 12 TPI). If the override switch is set to any other value but 100%, the thread will be stripped at best, and the tap broken at worst with related part damage.

The other problem will become evident only in a single block mode run, during setup or machining. Look at blocks N46 and N47. In the N46 block, tap reaches the end of

Z-axis - *while the spindle is still rotating!* True, the spindle will be stopped in block N47, but in single block mode it will be too late. A similar situation will happen during the feed-out motion. Spindle reverses in block N48 but does not move until block N49 is processed. Therefore, the program O2607 is a *very poor* example of tapping on CNC lathes.

These are some details usually not considered for a fixed cycle application (such as the G84 tapping cycle), when used for milling programs. For milling, all tool motions are built-in, so they are contained *within* the fixed cycle. To eliminate the first potential problem of feed rate override setting, programming M48/M49 functions will temporarily disable the feed rate override switch. Even better way is to replace the feed-in and feed-out tap motion command from the current G01 mode to G32 mode (G33 on some controls). G32 command is normally used for single point threading. Two major results will be achieved with G32 command - the *spindle will be synchronized*, and the feed rate override will be *ineffective* by default (automatically). That solves the first problem. The second problem will be solved if the spindle M-functions are programmed in the *same block* as the tool motion. That means joining the block N46 with N47, and block N48 with N49.

This much improved version of the tapping example is in a new program - O2608:

O2608 (TAPPING ON LATHES) (CORRECT VERSION IN PRACTICE) ... (T02 - TAP DRILL 31/64) ...

...

N42 M01

(T03 - 9/16-12 PLUG TAP)

N43 T0300 M42

N44 G97 S450 M03

N45 G00 X0 Z0.5 M08 T0303

N46 G32 Z-0.875 F0.0833 M05

N47 Z0.5 M04

N48 M05

N49 G00 X12.0 Z2.0 T0300 M09

N50 M30

%

The block (N48 in the example) containing spindle stop function M05, is not required if the tap is the *last* tool programmed, although it does no harm in any other program. Compare this newer program O2608 with program O2607. Program O2608 is a great deal more stable, and the possibility of any significant problem is virtually eliminated.

E. OTHER OPERATIONS

There are many other programming variations relating to machining holes on CNC machining centers and lathes. This chapter has covered some of the most important and most common possibilities.

Some fewer common applications, such as machining operations using tools for back boring, or block boring tools, tools with multiple cutting edges and other special tools for machining holes may be quite infrequent in programming. However, programming these *unusual* operations is no more difficult than programming the ordinary everyday tool motions, using ordinary everyday tools.

The real ability of a CNC programmer is measured in terms of applying the past knowledge and

experience to a new situation. It requires a thinking process, and it requires a degree of ingenuity and hard work.

24. CIRCULARINTERPOLATION

In the majority of CNC programming applications, there are only two types of tool motions related to contouring. One is *Linear Interpolation*, discussed earlier, the other one is *Circular Interpolation*, covered in this chapter. Programming method of controlling a toolpath along an arc is similar to the method of programming a toolpath along a line. The method of circular contouring is called *circular interpolation*. It is commonly used in profiling on CNC vertical and horizontal machining centers, as well as on lathes and many other CNC machines, such as simple milling machines, routers, burners, water jet and laser profilers, wire EDM, and others.

Circular interpolation is used for programming arcs or complete circles in such applications as outside and inside radiuses (blend and partial), circular pockets, spherical or conical shapes, radial recesses, grooves, corner breaks, helical cutting, even large counterbores, etc. CNC unit will interpolate a defined arc with a very high precision, if the necessary information is given in the program.

A. ELEMENTS OF A CIRCLE

To understand the principles of programming various circular motions, it helps to know something about the basic geometrical entity known as the *circle*. As an entity that is quite common in everyday life, a circle has various properties that are strictly mathematical, only considered in specialized disciplines, such as Computerized Numerical Control, motion control and automation.

The following definition of a circle and several other definitions that are related to a circle are based on some common dictionary definitions - *Figure 29-1*.

A circle is defined as a closed curve on a plane, where all points have the same distance from an

internal point called the circle center point

There are other similar definitions of a circle that can be found in dictionaries and mathematical books. General understanding of a circle and its various properties as described in this digital book, provides sufficient knowledge for most aspects of CNC programming. Additional knowledge will be needed for some specialized or complex programming applications. At this time, become at least reasonably familiar with basic geometrical and trigonometric relationships for arcs and circles.

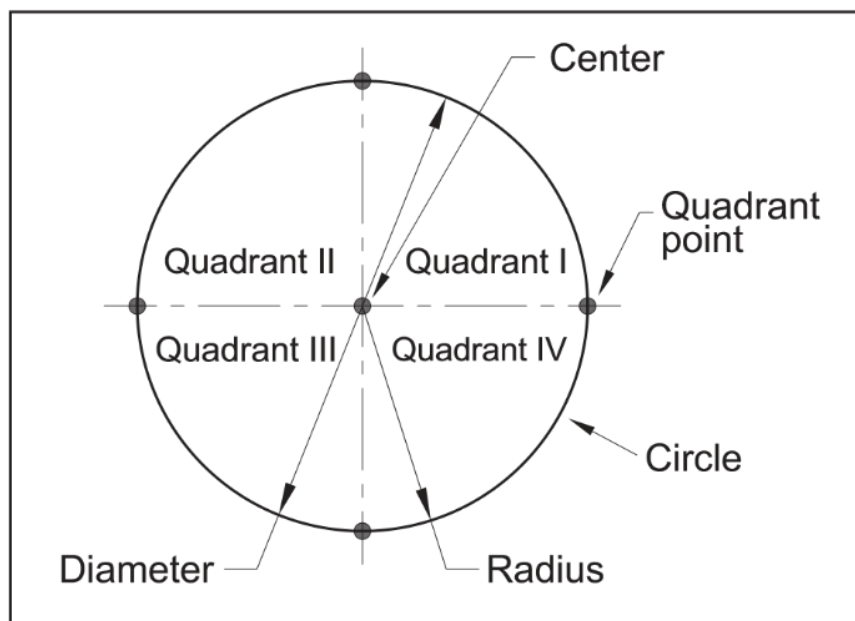


Figure 29-1

Basic elements of a circle

Radius and Diameter

In the simplest mathematical terms, a circle is defined by its center point and its radius. Two of the most important elements of a circle used in part programming are the circle *radius* and the circle *diameter*.

RADIUS of a circle is the line segment from the center point to any point on the circle

DIAMETER of a circle is the line segment through the center point of the circle and

having both end points on the circle

Center point location of the circle is also important for CNC programming. The plural form of the word *radius* is *radii*, although the word '*radiuses*' has been accepted as a colloquial term and is considered correct. In CNC programming, radiuses and diameters are used all the time, on a daily basis for almost all contouring machines. Drawings used in machine shops use radius and diameter dimensions for even the simplest parts, with an almost unlimited number of possible applications.

Radiuses and diameters are also used in relation to the cutting tool insert designation, they are used for measuring and gauging (inspections), as well as in trigonometric calculations and various auxiliary sketches. In programming, the actual application of an arc or circle is not important, only its mathematical characteristics.

Circle Area and Circumference

The *area* of a circle is defined by this formula:

A	R ²

+ where ...

$$A = \text{Area of the circle} \quad R = \text{Circle radius}$$
$$= \text{Constant (3.1415927)}$$

The *circumference* of a circle is the *length* of a circle if it were a straight line:

C	D

+ where ...

$$C = \text{Circumference of the circle} \quad D = \text{Circle diameter}$$
$$= \text{Constant (3.1415927)}$$

It is important to note that both the area and circumference of a circle (its actual length) are seldom used in CNC programming, although understanding their concepts presents a rather useful knowledge.

B. QUADRANTS

A *quadrant* is a major property of a circle and can be defined mathematically:

A quadrant is any one of the four parts of the plane formed by the system of rectangular coordinates

It is to every programmer's benefit to understand the concept of quadrants and their applications for circular motions in milling and turning programs.

A circle is programmed in all four quadrants, due to its nature, while most arcs are programmed within one or two quadrants. When programming the arc vectors, I, J and K (described later), the angular difference between the arc start and end points is irrelevant. The only purpose of arc vectors is to define a *unique* arc radius between two points.

For many arcs programming projects, the direct radius can be used with the R-address, available for majority of control systems. In this case, the angular difference between the start and end points is very important, because the computer will do its own calculations to find the arc center. Arc with the angular difference of *180 or less*, measured between the start and end points, uses an *R-positive* amount. Arc in which the angular difference is *more than 180°*, uses an *R-negative* amount. There are two possible choices, and the radius amount alone cannot define a unique arc.

Also worth mentioning is a *mirrored* toolpath and its relationship to the quadrants. Although it is not a subject of the current chapter, mirroring and quadrants must be considered together. What happens to the toolpath when it is mirrored is determined by the quadrant where the mirrored toolpath is positioned. Details about mirror image as a programming subject start on *page 409*. For now, it should be adequate to cover a very brief overview only.

For example, if a programmed toolpath in *Quadrant I* is mirrored to *Quadrants II* or *IV*, the cutting method will be reversed. That means a *climb* milling will become *conventional* milling and vice versa. The same rule applies to a programmed toolpath in *Quadrant II* as it relates to *Quadrants I* and *III*. This is a very important consideration for many materials used in CNC machining, because climb milling in *Quadrant I* will turn into conventional milling in *Quadrants II* and *IV* - a situation that is not always desirable. Similar changes will occur for other quadrants.

Quadrant Points

From the earlier definition should be clear that quadrants consist of two perpendicular lines that converge at the arc center point and an arc that is exactly one quarter of a circle circumference. In order to understand the subject deeper, draw a line from the center of an arc that is parallel to one of the axes and is longer than the arc radius. The line created an *intersection* point between the line and arc. This point has a special significance in part programming. It is often known as the *Quadrant Point* - or the *Cardinal Point* - although the latter term is not used very often, except in some mathematical applications. There are four quadrant points on a given circle, or four intersections of the circle with its axes. Quadrant points locations can be remembered easier by associating them with the dial of a common compass or a standard watch with an analog dial:

Degrees	Compass direction	Watch direction	Located between quadrants
0	EAST	3 o'clock	IV and I
90	NORTH	12 o'clock	I and II
180	WEST	9 o'clock	II and III
270	SOUTH	6 o'clock	III and IV

At this point of learning, it may be a good idea to refresh some terms relating to angle *direction* definition. Established industry standard (mathematics, as well as CAD, CAM and CNC) defines an absolute angular value as being *positive* in the *counterclockwise* direction and *always starting from zero degrees*. From the above table, zero degrees correspond to the *East* direction or *three o'clock* position of an analog clock - *Figure 29-2*.

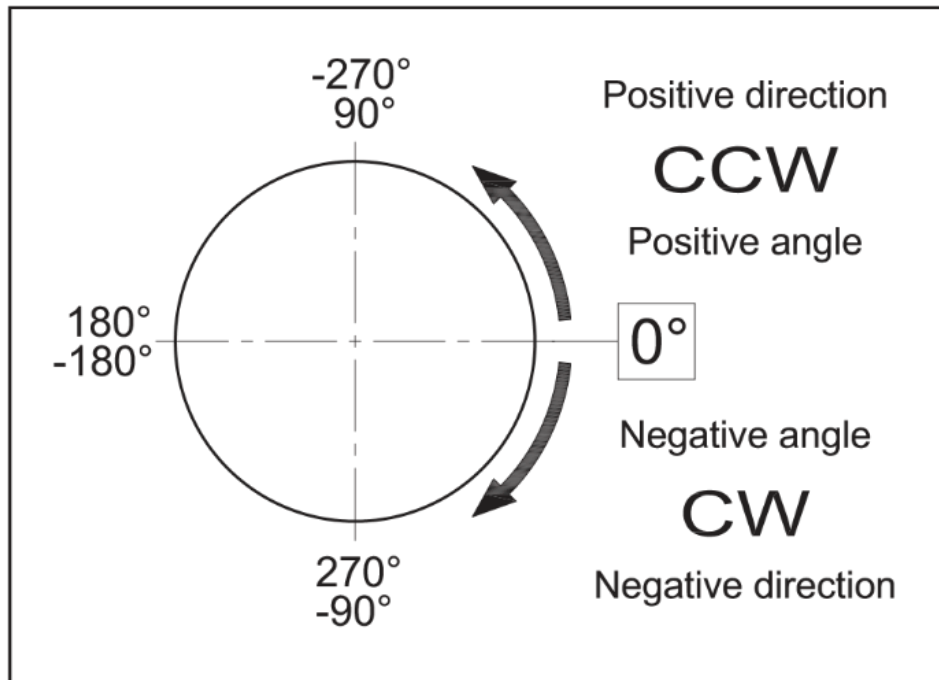


Figure 29-2

Mathematical definition of arc direction

There is another reason why quadrant points are important in CNC programming. In some cases, one or more quadrant points will be used as the arc *end points*, even if the circular cut covers more than one quadrant. This is particularly true for many older control systems, where crossing the quadrants in a single block is not allowed. Modern controls can generate arc of any length in a single block, up to 360, with very few restrictions.

C. PROGRAMMING FORMAT

Programming format for a circular interpolation toolpath must include several specifications, without which the task of cutting an arc would be impossible. The most important specifications are defined as:

Arc cutting direction (CW or CCW)

Arc start and end points

Arc center and radius value

Cutting feed rate must also be in effect, discussed in more detail later in this chapter. Special modal G-codes are used for circular motion programming and additional specifications related to the circle radius are also required.

Arc Cutting Direction

A cutting tool may move along an arc in two directions *clockwise* (CW) or *counterclockwise* (CCW). These two terms are assigned by convention. On most machines, the motion direction is determined by looking perpendicularly at the plane in which the circular motion is programmed. The motion *from* the plane vertical axis *towards* the plane horizontal axis is clockwise, reverse motion is counterclockwise. This convention has *mathematical* origins and does not always match machine axes orientation. Machining in planes is described as a separate subject, starting on *page 279*; this chapter will only take a brief look.

In a typical program format, the first statement of a circular cutting motion block is the *cutting direction*. This direction defines cutting tool motion along the programmed arc, which is either clockwise or counterclockwise. The motion direction along an arc is programmed using one of two preparatory commands in the block.

Circular Interpolation Block

All controls offer two preparatory commands associated with programming an arc direction:

G02	Circular motion clockwise	<i>CW</i>
G03	Circular motion counterclockwise	<i>CCW</i>

Both the G02 and G03 commands are modal, therefore they remain in effect until the end of program or until canceled by another command from the same G-code group, usually by another motion command.

Preparatory commands G02 and G03 are the key words used in programming to establish circular interpolation mode. Coordinate words following the G02 or G03 command are always designated within a selected plane. The plane is normally based on the available axis's combinations of XY, ZX and YZ for milling or similar applications. Normally, there is no plane selection on a lathe, although some control indicates it as G18, the ZX plane.

Plane selection and the combination of circular motion parameters with the arc cutting direction determine the *arc end point*, and the R-amount specifies the *arc radius*. Special arc center *vectors* (also known as *modifiers*) are also available, if the programmer requires them.

When G02 or G03 command is activated by a CNC program, any currently active tool motion command is automatically canceled. This canceling motion is typically G00, G01 or a cycle command. All circular toolpath motions must be programmed with a cutting feed rate in effect, applying the same basic rules as for linear interpolation. That means the feed rate F must be programmed *before* or *within* the cutting motion block. If feed rate is not specified in the circular motion block, control system will automatically search for the *last* programmed feed rate. If there is no feed rate in effect at all, many controls usually return an error message (an alarm) to that effect. Cutting feed rate may be specified in one of two ways. Either directly, within the arc cutting block only or indirectly, by assuming the last active feed rate. Circular motion in rapid mode is generally not possible. Also not possible is a simultaneous three axes *circular* motion. For more details on this subject, look up the subject describing helical milling (*see page 433*).

On the majority of older controls, direct radius address R cannot be specified, and the arc center vectors I, J and K must used instead:

G02 X.. Y.. I.. J..	Milling program - CW
G02 X.. Z.. I.. K..	Turning program - CW
G03 X.. Y.. I.. J..	Milling program - CCW
G03 X.. Z.. I.. K..	Turning program - CCW

Control systems supporting the arc radius designation by address R will *also* accept the IJK vectors, but the reverse is *not* true. If *both* arc vectors IJK and radius R are used in the same block, the *radius* amount takes priority, *regardless of the order of words*:

G02 (G03) X.. Y.. R.. I.. J..
G02 (G03) X.. Y.. I.. J.. R..

Controls that accept *only* the vectors IJK will return an error message in case the circular interpolation block contains R-address (an unknown address).

Arc Start and End Points

The *start point* of an arc is the point where circular interpolation *begins*, as determined by the cutting direction. This point *must* be located on the arc, and it will always be an endpoint - tangency point or an intersection - resulting in a blend radius or a partial radius respectively. The instruction contained in the start point block is sometimes called the *departure command* - *Figure 29-3*.

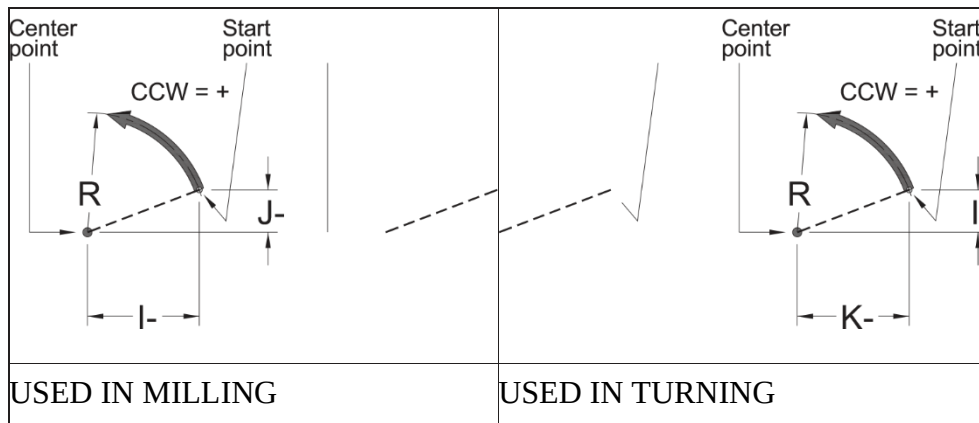


Figure 29-3

Center point and start point of an arc

Arc start point is always relative to the cutting motion direction and is represented in the program by coordinates in the block *preceding* the circular motion. In terms of a definition,

Start point of an arc is the last position of cutting tool before circular interpolation block

Here is an example:

N66 G01 X5.75 Y7.5 (START POINT OF ARC) N67 G03 X4.625 Y8.625 R1.125 (ARC) N68 G01 X. Y..

In the example, block N66 represents the *end* of a con tour, such as a linear motion. It also represents the *start* of the arc that follows next. In the following block N67, the arc is machined, so the coordinates represent the end of arc and start point of the next toolpath element. The last example block N68 represents the end point of the element that started from the arc. The *end point* of the arc is the coordinate point of any two axes, where the circular motion *ends*. This point is sometimes called the *target* position.

Arc Center and Radius

The *radius* of an arc can be designated with the address R or with arc center vectors I, J and K. The R-address allows programming the arc radius directly, IJK arc center vectors are used to actually define the physical (actual) arc center position. Most modern control systems support R-address input, older controls require the arc center vectors only. Basic programming format will vary only slightly between milling and turning systems, particularly when the R-ad-

dress version is used:

G02 X.. Y.. R..

Milling program - CW

G02 X.. Z.. R..

Turning program - CW

G03 X.. Y.. R..

Milling program - CCW

G03 X.. Z.. R..

Turning program - CCW

Why is the arc center location or the arc radius needed at all? It would seem that the end point of an arc programmed in combination with a circular interpolation mode should be sufficient. This is never true. Always keep in mind that *numerical control* means control of the toolpath by *numbers*. In this case, there is an *infinite number* of mathematical possibilities where all may correspond to this incomplete definition. There is virtually an unlimited number of arc radiuses that will fit between the programmed start and end points and still maintain the cutting direction.

Another important concept to understand is that the cutting direction CW or CCW has nothing to do with the arc center or the radius. Control system needs more information than direction and target point in order to cut the desired arc. This additional information must contain a definition that defines a programmed arc with a *unique* radius.

This unique radius is achieved by using the R-address for direct radius input or using the IJK arc center vectors. Address R is the actual radius of the toolpath, usually the radius taken from part drawing.

Arc Center Vectors

Figure 29-4 shows the signs of arc vectors I and J in all possible orientations. In different planes, different pairs of vectors are used, but the logic of their usage remains exactly the same.

Arc vectors I, J and K are used according to the following definitions (only I and J are shown in the illustration):

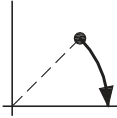
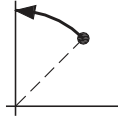
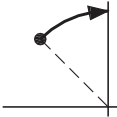
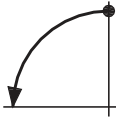
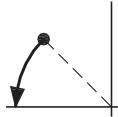
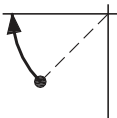
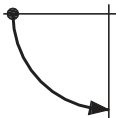
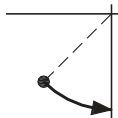
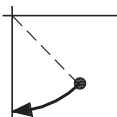
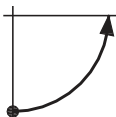
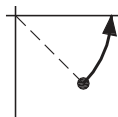
	G02		G03	
Quadrant I				
	I0 J-	I- J-	I- J0	I- J-
Quadrant II				
	I+ J0	I+ J-	I0 J-	I+ J-
Quadrant III				
	I0 J+	I+ J+	I+ J0	I+ J+
Quadrant IV				
	I- J0	I- J+	I0 J+	I- J+

Figure 29-4

Arc vectors I and J (also known as arc modifiers) and their sign designation in different quadrants of plane shown)

Arc center vector *I* is the distance, with specified direction, measured from the start point of the arc,

to the center of the arc, parallel to the X-axis

Arc center vector *J* is the distance, with specified direction, measured from the start point of the arc,

to the center of the arc, parallel to the Y-axis

Arc center vector *K* is the distance, with specified direction, measured from the start point of the arc,

to the center of the arc, parallel to the Z-axis

The distance between the arc start point and the arc center point (as specified by IJK vectors) is almost always measured as an *incremental* distance between the two points. Some control systems, for example some older Cincinnati designs, use the absolute designation to define an arc center. In those cases, the arc center is programmed as an *absolute* value from *program zero*, not from the arc center. Always make sure how each of the control systems in the machine shop handles these situations. Generally, it can be said that:

Arc center is measured *from* arc start point *to* arc center

The lack of a suitable standard in this respect creates a major difference in programming format, so be careful to avoid a possible error. An error can be particularly likely in those cases where both types of controls are installed in the machine shop. There is *no compatibility* between programs using absolute and incremental method of the arc center.

The specified *direction* applies only to the incremental designation of arc center. It is the definition of relative position of the arc center from the start point, programmed with a directional sign - absence of the sign always assumes a positive direction, minus sign indicates a negative direction and must always be written. Arcs using absolute center definition follow standard rules of absolute dimensioning.

Arc in Planes

For machining centers, programming an arc in any one of the three geometrical planes is allowed - *Figure 29-5*. The correct arc vectors must be used for each selected plane:

G17 G02 (G03) X.. Y.. R.. (or I.. J..)

G18 G02 (G03) X.. Z.. R.. (or I.. K..)

G19 G02 (G03) Y.. Z.. R.. (or J.. K..)

Arc cutting direction in three planes - the axis orientation is based on mathematical - not machine - planes

If the programmed plane is not aligned with the machine axes, or if the axes used in the program are selected without a plane designation, circular motion will take place according to the *axis selection in the program*. Always watch when the modal axis motion is omitted. The safest method of avoiding this potentially harmful problem is to follow a simple precaution - *never count on modal settings*.

In nonstandard planes, the circular program block should always contain specifications for *both* axes, as well as both arc vectors *or* the R radius. Such a block is *complete* and will always be executed on the basis of axes designation priority. This method is preferable to the selection of a previously defined plane. Even if the plane designation is incorrect, the resulting tool motion *will* always be correct.

D. RADIUS PROGRAMMING

Programming arc is very common. By definition, an arc is only a *portion* of a circle and there are many ways to program an arc. If the arc is 360, it must be programmed with the cutting start position being the same as its end position. In this case, a full circle is the result. If only a portion of the circle is programmed, only a radius is programmed. Two kinds of radii are used in CNC programming:

Blend radius

Partial radius

Each radius may be programmed in CW or CCW direction and each may be external or internal,

as well as in any orientation that the cutting tool can handle.

Blend Radius

A point of tangency between an arc and its adjacent element creates a *blend* radius. Blend radius is defined as a radius tangent between a *line and an arc*, an *arc and a line*, or between *two arcs*. A blend arc creates a smooth transition between one contour element and another. The point of tangency is the only contact point between the two elements.

The simplest form of a blend radius is between two perpendicular lines that are parallel to the machine axes. Calculation of the start and end points requires only a few additions or subtractions. A little more complex calculations are required when even one line is at an angle. In this case, trigonometric functions are used to calculate the start or the end point, or both. Similar calculations are required for blends between other entities as well. A blend arc is also known as a *fillet arc* or a *fillet radius*.

Partial Radius

The opposite of a blend arc is a *partial* arc - there is no smooth blend between two contour elements, instead, there is an intersection. Mathematically, there are always two possible selections, however, the part drawing should be quite clear as to the shape of any partial radius. Partial radius can also exist between *two lines*, *one line and an arc*, or between *two arcs*. Partial radius can be defined as a radius where either the start point, or the end point is *not* tangent to the adjacent element but intersects it in two places. Actual calculation of point coordinates for the arc start or end point is about the same as that for a blend arc, depending which method of dimensioning had been used in the part drawing.

E. ARC PROGRAMMING

With a full arc cutting, which means the *complete* 360° motion, the R-address cannot be used at all. Arc center vectors I and J must be applied, even on latest controls.

What if the circle is 359.999? Well, at first, circle *must* have 360°, therefore the word 'circle' is incorrect. Even a small difference of 0.001° does make a difference between a circle and an arc. Although this difference is much more important mathematically than for practical programming, the distinction is very important. In circular interpolation terms, an *incomplete circle* is nothing more than an *arc*.

Look at this arc a little differently. If a 90 arc is made, the R-address (direct radius) can be programmed, for example:

G01 X2.0 Y5.25 F12.0

G02 X3.75 Y7.0 R1.75

If an arc that covers *exactly* 180 is programmed, the program will not be much different:

G01 X2.0 Y5.25 F12.0

G02 X5.5 Y5.25 R1.75

Note that the Y-coordinate is the same for the arc start and end position. Y amount in the circular motion block does not have to be repeated, it is used here only for illustration.

Another example shows programming an arc of 270°, still using the R-address. Are the following blocks correct?

G01 X10.5 Y8.625 F17.0

G02 X13.125 Y6.0 R2.625

The blocks appear to be correct. All calculations, format, individual words, they all appear to be right. Yet, the program is *wrong*! Its result will be a 90° arc, not 270°.

Study the illustration in *Figure 29-15*. It shows that there is not just one, but two mathematical possibilities when the R-address is used for arcs. The solid contour shown is the toolpath, dashes identify the two possible radiuses.

Programmers do not normally think of these mathematical alternatives, until they program arcs *larger* than 180° (or scrap a part). This is a similar situation to that of a full circle, described earlier. Although the I and J vectors can be used to remedy this problem, a different remedy may be a preferred choice. R-address can still be used in the program, but with a *negative* sign for any arc that is greater than 180°. For arcs smaller than 180°, the usual positive R radius remains in effect. Recall from some earlier explanations that if there is no sign with the R word (or any other word), the word assumes a *positive* value. Compare the two programming examples:

G01 X10.5 Y8.625 F17.0

G02 X13.125 Y6.0 R2.625 (90 DEGREES)

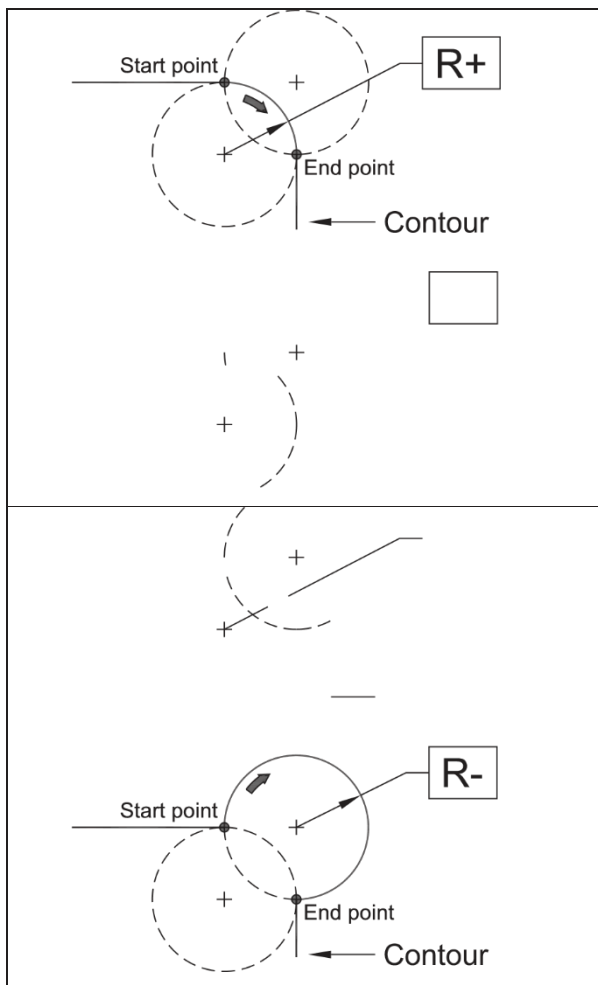


Figure 29-15

Sign of R address for circular cutting - only the center is different

The following example is identical to the previous one, except for the R-address sign.

G01 X10.5 Y8.625 F17.0

G02 X13.125 Y6.0 R-2.625 (270 DEGREES)

If frequently programming arcs that cover more than

180, establish a particular programming style. If the style is well thought out, it will avoid the costly mistakes associated with the R-address sign error.

25. TOOL NOSE RADIUS OFFSET

All general principles and rules described so far also apply to the radius offset for a lathe contouring tool. There are few differences, mainly caused by the shape of the tool.

In milling, the cutting tool is always round. The cutter circumference (periphery) is the cutting edge, and its radius is the nominal offset. Turning tools have a different design. The most common is a multi-sided carbide insert. An insert may have one or more cutting edges. For strength and longer insert life, the cutting edge has a relatively small corner radius. Typical radiuses for turning and boring tools are:

0.40 mm	(metric)	or	$1/64 = 0.0156$	(imperial)
0.80 mm	(metric)	or	$1/32 = 0.0313$	(imperial)
1.20 mm	(metric)	or	$3/64 = 0.0469$	(imperial)

Because lathe tool cutting edge is often called a *tool nose*, the term *tool nose radius offset* became common.

A. TOOL NOSE TIP

Actual tool nose is usually a corner of the tool, where two cutting edges blend into a nose radius. Figure 30-32 shows typical corners of a turning tool and a boring tool.

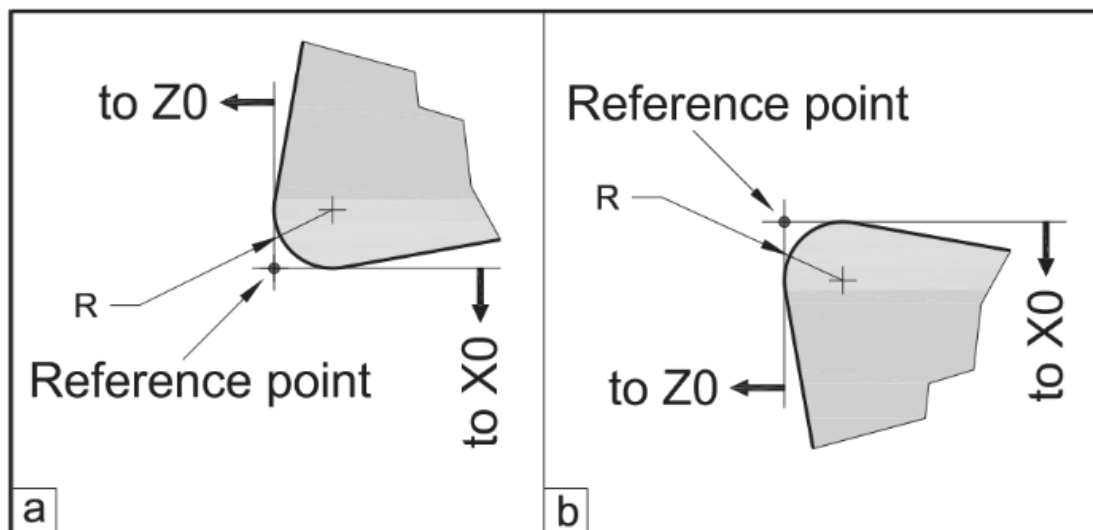


Figure 30-32

Tool reference point for typical turning (a) and boring (b) - rear lathe

The tool nose *reference point* in turning is often called the *command point*, *imaginary point* and lately, even the *virtual point*. It is *this* point that is moved along the contour, because it is directly related to X0Z0 setting of the part.

B. RADIUS OFFSET COMMANDS

The same preparatory commands relating to cutter radius used in milling operations are also used for contouring on CNC lathes - Figure 30-33:

G41	Offset of tool nose radius to the LEFT of contouring direction
G42	Offset of tool nose radius to the RIGHT of contouring direction

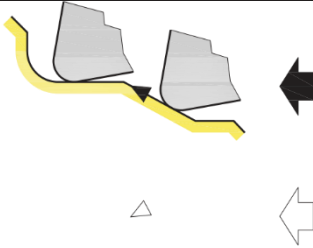
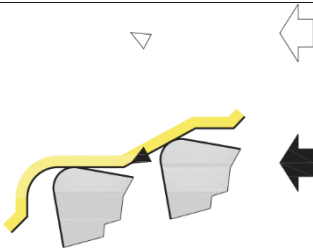
G40	Offset of tool nose radius ANCEL
RIGHT	
LEFT	

Figure 30-33

Tool nose radius offset applied to OD (G42) and ID (G41)

There is one major difference - for lathes, G-codes do not use the D-address - all actual offset values are stored in the *Geometry/Wear* offset. *Lathe tools have different cutting edges*, otherwise they are similar to milling.

C. TOOL TIP ORIENTATION

The center of a circle symbolizing an end mill must be equidistant to the contour by its radius. In milling, cutting edges are part of the tool radius, on lathes, they are not. Lathe tools *do* have a radius but separate cutting edges. The nose radius center is also equidistant from the contour, and the edges change their orientation, even for the same insert. Additional definitions are needed in a form of a vector pointing *from the command point towards the radius center*. This vector is called the *tool tip orientation*, arbitrarily numbered. Control system uses this number to establish the nose radius center and its orientation from the command point. *Figure 30-34* shows two tools and their tip orientation.

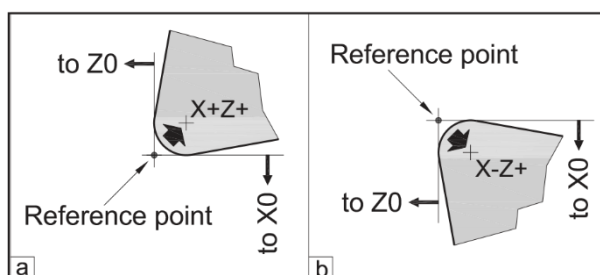


Figure 30-34

Relationship between tool nose reference point and radius center

Tool tip orientation is entered during the setup, according to some arbitrary rules. Fanuc controls require a fixed single digit number for each possible tool tip orientation. This number has to be entered into the offset screen at the control, under the T column heading. Value of the tool radius R must also be entered, as they both work together. If the tool tip is 0 (or 9), the control will compensate to the center. *Figures 30-35 and 30-36* show the standard tool tip numbering for *rear orientation* CNC lathes - those with X+ up and Z+ to the right of origin.

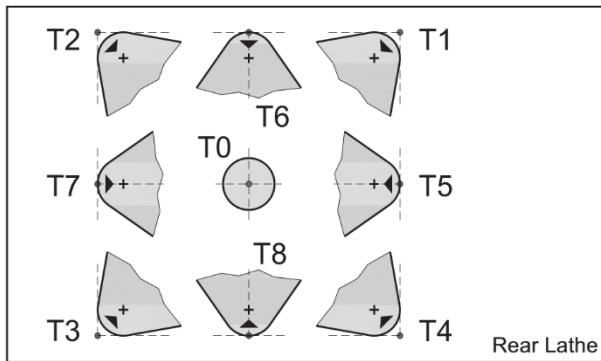


Figure 30-35

Arbitrary tool tip numbers for tool nose radius offset orientation

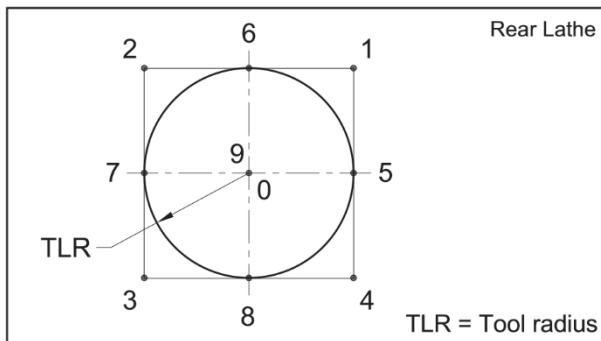
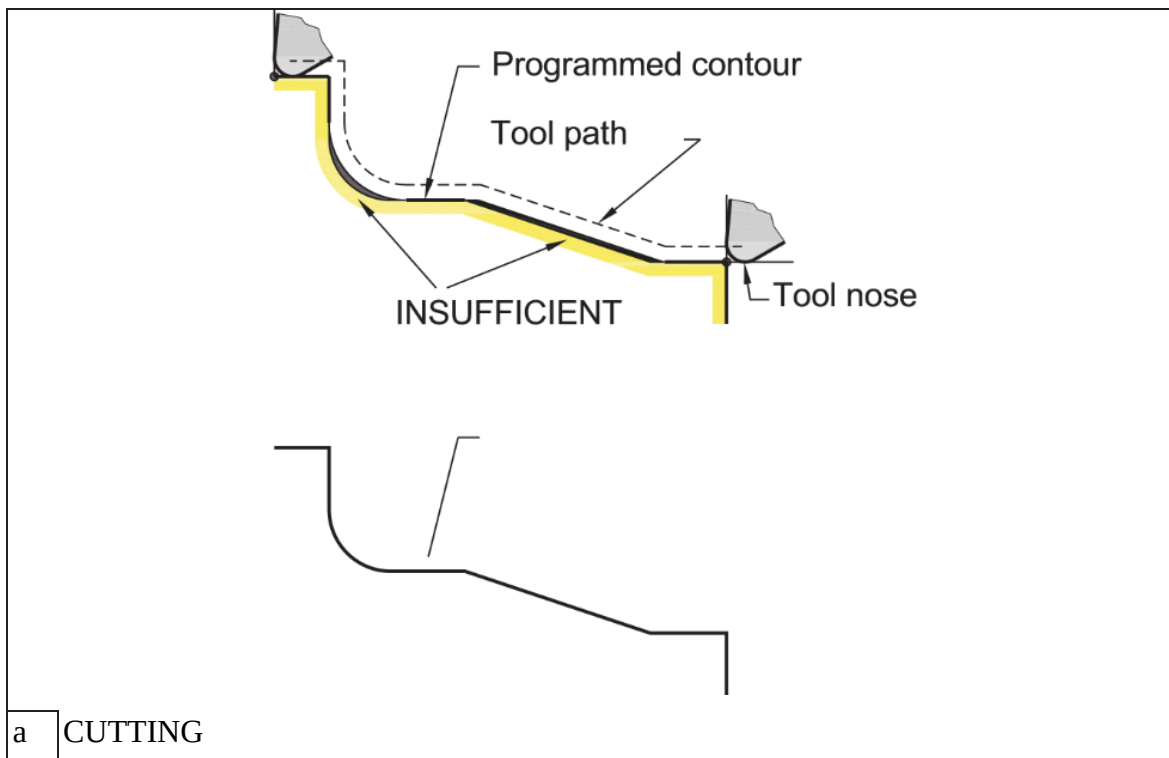


Figure 30-36

Schematic illustration of Fanuc type tool tip numbering

D. EFFECT OF TOOL NOSE RADIUS OFFSET

Some programmers do not bother using the tool nose radius offset. *That is wrong!* Study Figure 30-37 carefully first - explanations follow.



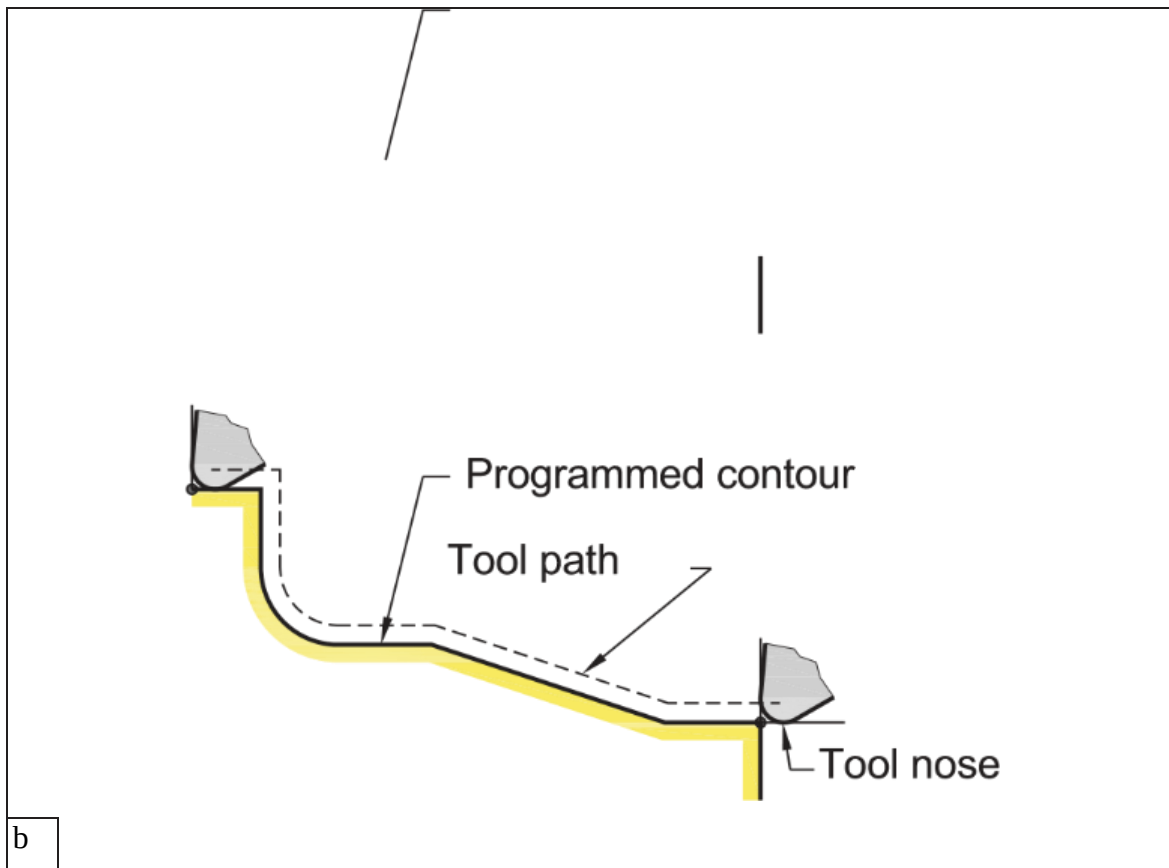


Figure 30-37

Effect of tool nose radius offset - (a) offset NOT used - (b) offset used

Theoretically, there is no need for the offset if only a single axis motion is programmed. However, single axis motions are typically part of a contour that also includes radii, chamfers and tapers. In such cases, tool nose radius offset is needed, otherwise all radii, chamfers and tapers will not be correct, and part will be scrap.

The last illustration shows what areas of part would be *undercut* or *overcut*, if tool nose radius offset were *not* used during machining. Note that negative effect applies to a two-axis simultaneous motion only.

E. SAMPLE PROGRAM

The following program example O3005 shows a simple application of tool nose radius offset on an external and internal contour, based on the drawing in Figure 30-38. Only the finishing cuts are shown - roughing is also necessary but would most likely use G71 multiple repetitive roughing cycle without radius offset.

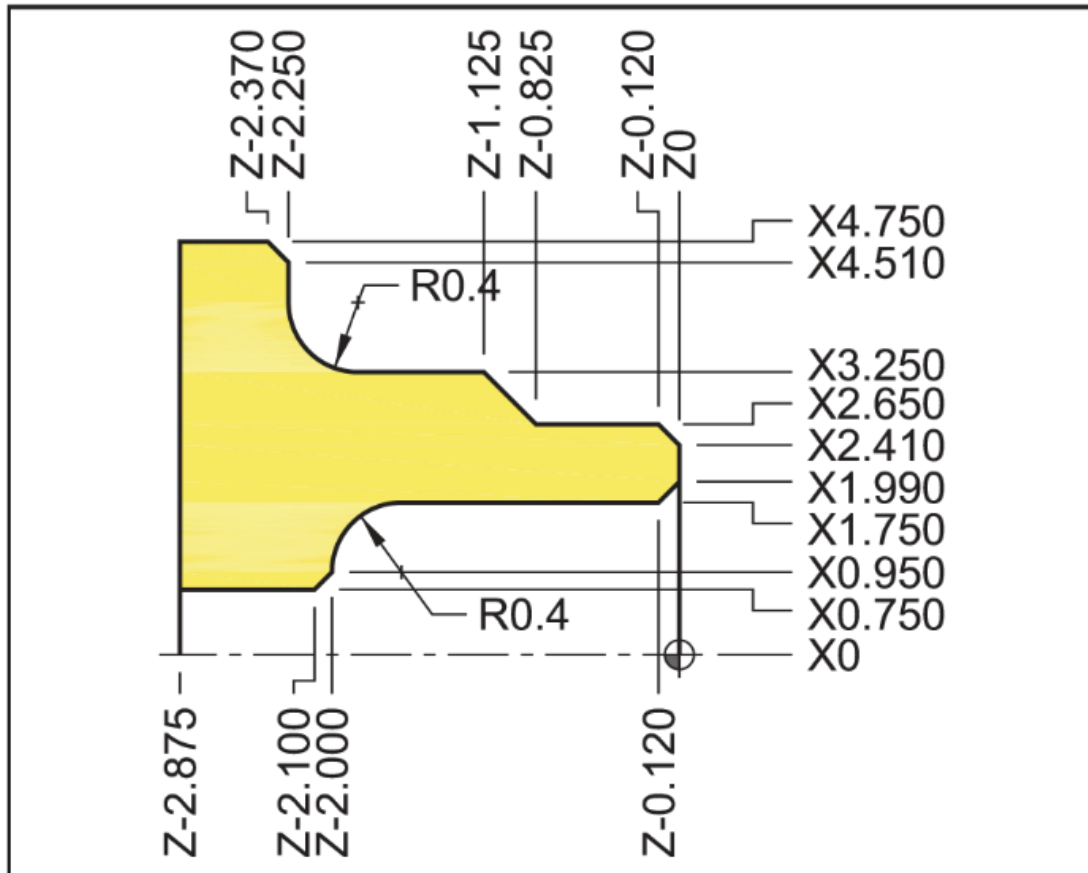


Figure 30-38

Simplified sample drawing for program example O3005

O3005 (G20) ...

N31 T0300 (EXTERNAL FINISHING)

N32 G96 S450 M03

N33 G00 G42 X2.21 Z0.1 T0303 M08

N34 G01 X2.65 Z-0.12 F0.007 N35 Z-0.825 F0.01

N36 X3.25 Z-1.125

N37 Z-1.85

N38 G02 X4.05 Z-2.25 R0.4 N39 G01 X4.51

N40 X4.8 Z-2.395

N41 U0.2

N42 G00 G40 X8.0 Z5.0 T0300

N43 M01

N44 T0400 (INTERNAL FINISHING)

N45 G96 S400 M03

N46 G00 G41 X2.19 Z0.1 T0404 M08

N47 G01 X1.75 Z-0.12 F0.006
 N48 Z-1.6 F0.009
 N49 G03 X0.95 Z-2.0 R0.4
 N50 G01 X0.75 Z-2.1
 N51 Z-2.925
 N52 U-0.2
 N53 G00 Z2.0
 N54 G00 G40 X8.0 Z2.0 T0400

—————
 N55 M01
 ...

Note that both contour start, and end positions are in a *clear area* - away from the part - for the same reason as in milling. Make sure to program sufficient clearance for both lead-in and lead-out motions. *Cutter radius compensation interference alarm* is always caused by insufficient clearance - cutter radius cannot fit into the space provided.

F. MINIMUM CLEARANCE REQUIRED

As a rule, each physical clearance in the program should be large enough to accommodate **double** tool nose radius

The rule should be clear. *Figure 30-39* shows minimum clearances when set at the start and end of cut. Make sure the nose radius fits into this clearance two or more times. Symbols $>TLR \ 2$ and 4 mean the clearance should be greater than twice or four times the nose radius. Double radius per side becomes a quadruple radius on a diameter.

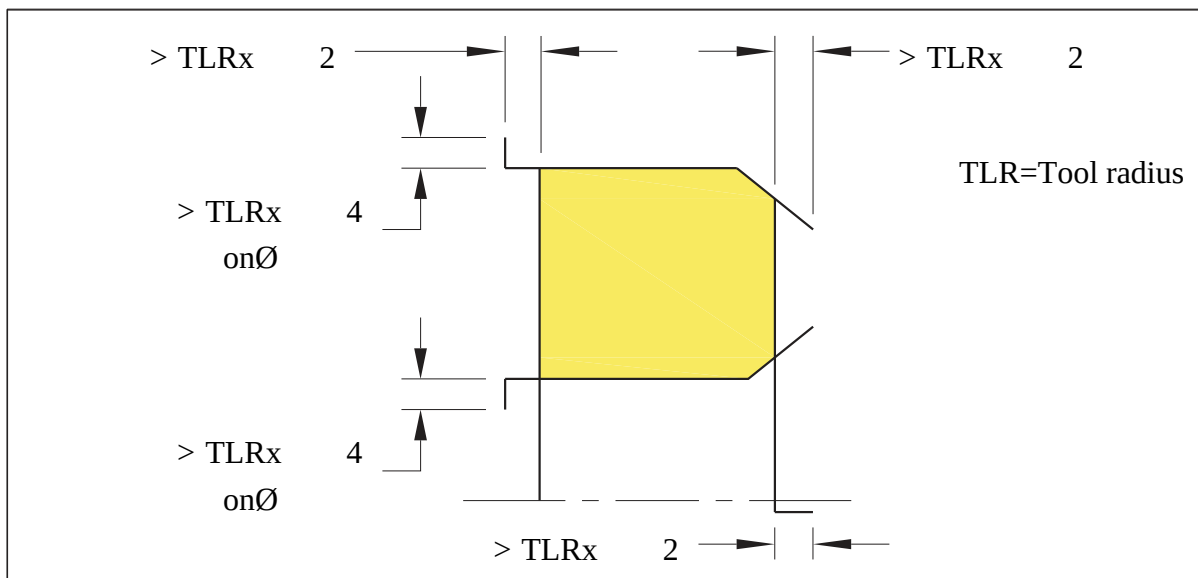


Figure 30-39

Minimum clearances for tool nose radius offset

	Nose radius	Per side	On diameter
mm	0.4	0.8	1.6

N23 G41 X25.0 Z3.0 T0707 M08 (START POS.)
N24 G01 Z-30.0 F0.25 (NOT SHOWN ON DWG)
N25 U-5.0 (==> CRITICAL RETRACT AMOUNT)

N26 G00 Z3.0 (OUT OF HOLE FIRST)
N27 G40 X200.0 Z50.0 T0700 (THEN CLEAR+CANCEL) N28 M01

The critical block is N25 - this is where most errors happen. Many programmers many think that 5 mm clearance is too large, and the boring bar may hit the opposite part of the hole. First, 5 mm is on *diameter*, so it is only 2.5 mm per side. Second, into this 2.5 mm clearance a radius of 0.8 mm must fit - *twice!* That takes away another 1.6 mm from 2.5 mm, making the actual amount of tool motion only 0.9 mm (0.035"). The reason is that the tool radius has to be to the left of *all* motions, not just for the actual bore.

H. CHANGE OF MOTION DIRECTION

The second common problem is when facing in tool nose radius offset mode. On CNC lathes, a change in cutting direction is used much more often than on machining centers.

The following example shows a facing cut on a solid face with G41 in effect, changing to a turning cut(-s) with G42 in effect - see *Figure 30-41*. Possible problem is discussed as well.

...

N21 T0100 (CORRECT APPROACH)
N22 G96 S400 M03
N23 G00 G41 X1.7 Z0 T0101 M08 (START)
N24 G01 X-0.07 F0.007 (FACE OFF)
N25 G00 Z0.1 (ONE AXIS ONLY)
N26 ~~G42~~ X1.0 (THEN COMPENSATION)
N27 G01 X1.4 Z-0.1 F0.012 (CONTOURING)
N28 Z-0.65 N29 X...

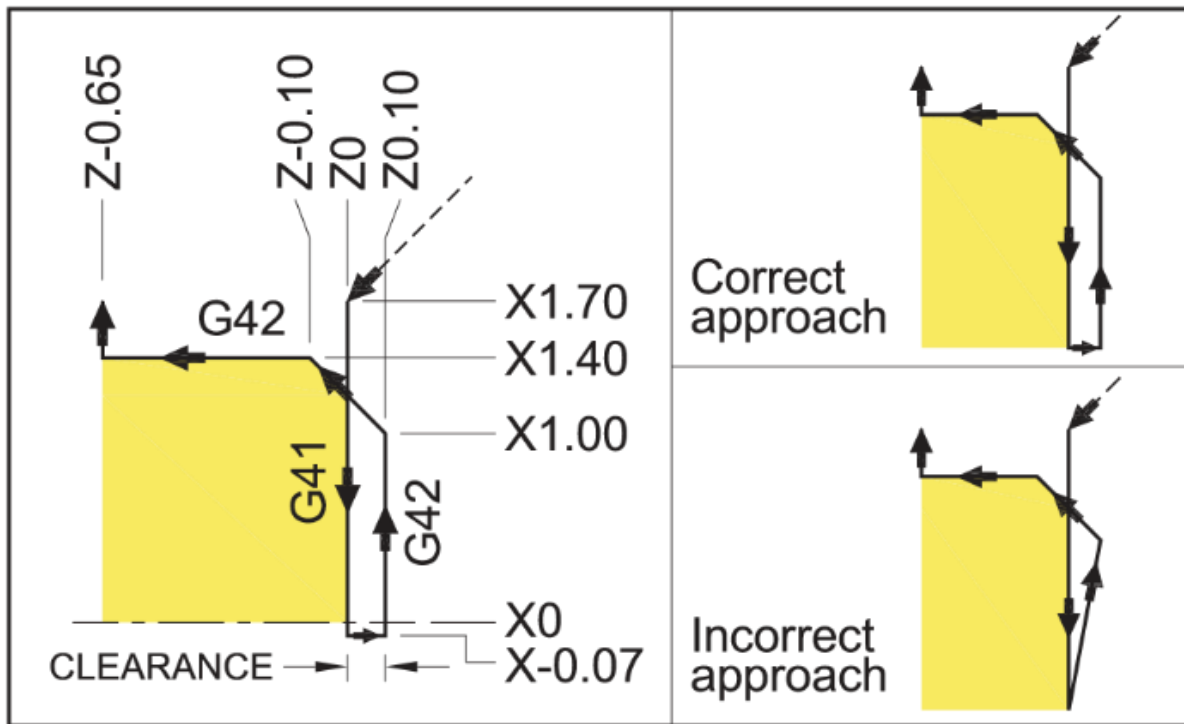


Figure 30-41

Tool nose radius offset change in facing for the same tool

Face cutting is a single axis motion, and the radius offset is used for consistency, or it may be in effect for another reason. For solid parts, the face cut must end *below* the center line - X-0.07 in block N24 - at a diameter marginally larger than double tool radius. If the cut finishes at X0, the tool leaves a small unfinished tip at the center line and the face will not be flat. Also compare the *correct* and *incorrect* tool motions on the right side of the last illustration. If the above program is modified to the following version,

...

N21 T0100 (INCORRECT APPROACH)

N22 G96 S400 M03

26. TURNING AND BORING

There is so much information that can be covered in this section, that a whole book could be written just about turning and boring. Selected subjects are presented in this chapter, others are covered in chapters dealing with lathe cycles, grooving, part-off, single point threading, etc.

A. TOOL FUNCTION - TURNING

In terms of distinction, *turning* and *boring* are practically identical operations, except for the area of metal removal where the actual machining takes place. Often, terms *external turning* and *internal turning* are also used, meaning the same as turning and boring respectively. From programming perspective, the rules are virtually the same, and all significant differences will be covered as necessary.

CNC lathes require programming the selected tool by its tool number, using the T-address. In comparison with a CNC machining center, the tool function for lathes is more extensive and calls for additional details. One major difference between milling and turning controls is the fact that the T-address for CNC lathes *will make the actual tool change*. This is not the case in milling. No M06 function exists on a standard CNC lathe.

T-Address

One difference from machining centers is that a tool defined as T01 in the program *must* be mounted in the turret station #1, tool defined as T12 must be mounted in turret station #12, etc. Another difference between milling and turning tools is in the *format* of the T-address. Format for turning system is T4, or more accurately, T2+2. The first two digits identify the turret station number and geometry offset; the last two digits identify the wear tool offset number for the selected tool station - *Figure 34-1*.

Txxyy format represents tool station xx and wear offset number yy. For example, T0202 will cause the turret to index to the tool station #2 (first two digits) which will become the working station (active tool). At the same time, the associated tool wear offset number (the second pair of digits) will become effective as well, unless it is 00.

Selection of tool number (the first pair of digits) also selects the geometry offset on most modern CNC lathes. In that case, the second pair of digits will select tool wear offset number. Any tool station selected by the turret station number identification can be associated with any offset number within the available offset range. In most applications, only one tool offset number is active for any selected tool. In such a case, it is wise to program the offset number the same as the tool number. Such an approach makes the operator's job much easier.

Consider the following choices:

G00 T0214 Tool station02, wearoffset14

G00 T1105 Tool station11, wearoffset05

G00 T0404 Tool station04, wearoffset04

Although all examples are technically correct, only the last example format is recommended. When many tools are used in a program, offset numbers for individual tools may be confusing, if they do not correspond to the tool station numbers. There is only one time when the offset number *cannot* be the same as the tool station number. That happens in those cases when *two or more offsets* are assigned to the same tool, for example T0202 for the first wear offset, T0222 for the second wear offset.

Leading zeros in the tool function can be omitted for the tool number selection, but not for selection of the wear offset number. T0202 has the same meaning when written as T202. Eliminating the leading zero for tool wear offset will

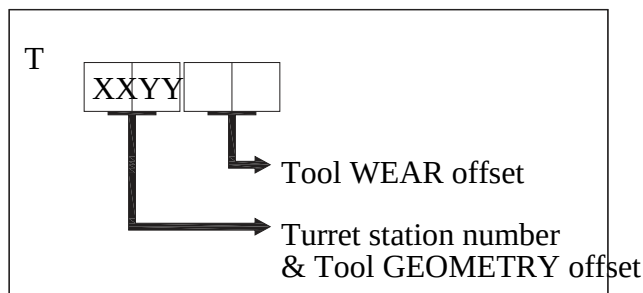


Figure 34-1

Typical tool function address for CNC lathes

result in an incorrect statement:

T22 means *T0022*, which is an illegal format.

In summary, the active side of the turret (tool station) is programmed by the first pair of digits, the wear offset number is programmed by the last pair of digits in the tool function command:

G00 T0404

The most useful preference is to disregard the leading zero suppression and use the tool function in its full format, as shown above and in all examples in this digital book.

B. LATHE OFFSETS

Although tool offset has been to some extent covered in the previous section describing the tool function, it is a very important feature for turning systems and some review will be beneficial.

Geometry offset is measured for *each tool* as the actual *distance from the tool reference point to the program zero* (Z-axis distance will be stored as a negative value and so will be the X-diameter) - Figure 34-2.

Geometry offset defines the tool position in relation to program zero

Wear offset is used for fine-tuning dimensions

The best way to illustrate the importance of tool wear offset, is to consider a program that does *not* use it. All programmed dimensions are ideal values, based on the drawing. Variable insert tolerances are not considered, neither is the tool wear. Any deviation from programmed dimensions caused by the actual tool size will produce an incorrect dimension when the part is machined, a very important concern for jobs with tight tolerances. Tool wear offset is used to 'fine tune' the actual machined dimensions against intended programmed dimensions.

The purpose of tool wear offset is to adjust any difference between the programmed dimensions and the actual tool position on the part contour. If the wear offset is not available to the control, all adjustments are made to the only offset available - that is to the geometry offset.

Offset Entry

Tool offset number can be entered into the program in two different ways:

As a command *independent* of the tool motion

As a command applied *simultaneously* with a tool motion statement

Independent Tool Offset

For an independent offset entry in the program, tool offset is applied *together with tool indexing*:

N34 G00 T0202

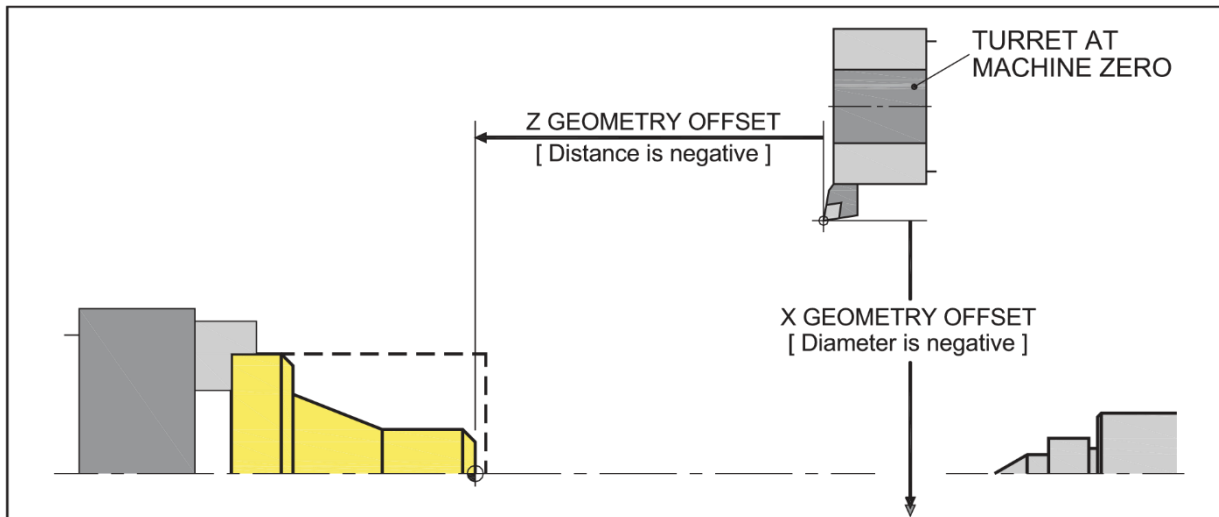


Figure 34-2

Geometry offset is the distance from tool reference point to program zero, measured along an axis in machine zero

This command is usually programmed as the first block for each tool (in a clear position). If the older G50 position register is used, the offset is programmed together with, or immediately following, the coordinate register block. At this point, the tool is still at its indexing position. When tool offset is activated, it will cause a *physical motion* by the amount stored in the offset register. Note the preparatory command G00 before the tool function. This is a very important command, since it will enable the physical offset motion to actually take place. G00 is more important for the first tool but should be programmed for any tool. On page 23, covering the control system, the initial control status (when the power is turned on) is described. Since the control system usually assumes the G01 command (linear interpolation) at the start up, feed rate would be required. However, it looks rather absurd to program T0202 F0.025, although it is correct. Rapid motion is far more practical and rather than depending on the current control status, programming the G00 command will *always* get the tool offset activated.

Tool Offset with Motion

The second method is to program the wear offset simultaneously with a cutting tool motion, usually during tool approach towards the part. This is the preferred method. The following two examples illustrate this recommended programming of the T-function for turning systems - offset is activated when the second pair of digits in a tool number call are equal to or larger than 01:

N1 G20 T0100

N2 G96 S300 M03

N3 G00 X.. Z.. T0101 M08

...

Tool change in the first block N1 uses no offset number (00) - just the tool number, that is also the geometry offset number. The offset is applied two blocks later in N3.

In most cases, it makes no difference, whether the offset is activated with or without a motion command. But some limitations are possible when programming the tool offset entry *without* a motion command. For example, if the wear offset value stored is unusually large and the tool starts from machine zero position, this type of programming may cause an overtravel condition.

Even in cases of a small offset value, there will always be a 'jump' motion of the turret when the offset is activated. Some programmers do not like this jumpy motion, although it will do no harm to the machine. In these cases, the best approach is to activate tool wear offset during the

first motion, usually as a rapid approach motion towards the part. One consideration is very important when tool wear offset is activated together with a motion. Earlier in this chapter was a comment that the lathe tool function is also a function causing tool indexing. Without a doubt, the one situation to avoid is a simultaneous tool indexing and tool motion - it may have dangerous consequences.

In most cases, the best approach is to start each tool with tool indexing only, *without* any wear offset:

N34 T0200 M42

This example will register the coordinate setting for tool number 2, it will also index tool number 2 into working position, but it will *not* activate any offset (T0200 means index for tool 2 without tool wear offset). Gear range function may be added as well, if required. Such a block will normally be followed by the selection of spindle speed, and rapid approach to the first position, close to part. That is the block where the tool wear offset will be activated - *on the way* towards the first position:

N34 T0200 M42

N35 G96 S190 M03

N36 G00 G41 X12.0 Z0 T0202 M08

N37 G01 X1.6 F0.008 ...

Also note that no G00 is required for a block containing tool indexing with zero wear offset entry. Main advantage of programming the tool offset simultaneously with a motion is the elimination of any jumpy motion; at the same time, no overtravel condition will result, even if the wear offset is unusually large. Wear offset amount will only *extend* or *shorten* the programmed rapid approach, depending on the actual offset amount stored.

Generally, the tool wear offset register number is entered *before* or *during* rapid approach motion.

Offset Change

Most lathe programs require one offset for each tool. In some cases, however, the program can benefit if two or even more offsets are assigned to the same tool. Needless to say, only one offset can be active at one time. The current offset can be changed to another offset for the same tool to achieve additional flexibility. This is useful mainly in those cases when individual diameters or shoulder lengths must be machined to exact tolerances. Any new offset must be programmed *without* a cancellation of the previous one. In fact, this is the preferable method for changing from one offset to another. Reasoning is simple - remember that any offset change serves a purpose only during actual cutting. Offset cancellation could be unsafe if programmed during cutting motion. This is a very important - *and largely unexplored programming technique* - that some detailed examples are justified.

C. MULTIPLE OFFSETS

A large number of jobs machined on CNC lathes requires very high precision. High precision often requires tolerance ranges as specified in the engineering drawing and these ranges may have quite a variety. Since a single offset per tool is often not enough to maintain these tolerances, two or more wear offsets are required for a single tool.

The following three examples are designed to present a complete understanding of the advanced subject covering multiple offsets. The same basic drawing will be used for all examples - only the tolerances will vary.

Project itself is very simple - program and machine three diameters as per drawing and maintain tolerances at the same time. One rule at the beginning - the program *will not use* the middle

tolerance of the X or Z value. This is an unfortunate practice that makes changes to the program much more difficult at a later time, if drawing tolerances are changed by engineers or designers.

In the drawings, the following tolerances can be found:

Tolerances only on the diameter

Tolerances only on the shoulders (faces)

Tolerances on the diameters and shoulders

General Approach

The tolerances in all three examples are for training purposes only and will be much smaller in reality. All chamfer tolerances are, and non-specified tolerances are 0.005. That will allow concentration on the project. Material is a 1.5-inch aluminum bar, and three tools are used:

T01	For the face and rough contour
T03	For the finishing of contour to size
T05	0.125 wide part-off tool

Individual programmer's skill will determine the final result - how many offsets to program and where to enter them within the program. At the machine, CNC operator must store the correct setting amounts for each offset. In all cases, the main objective will be to aim for the middle tolerance in machining, *not in programming*.

Diameter Tolerances

Drawing in Figure 34-3 shows the sample part with variable tolerances applied only on diameters.

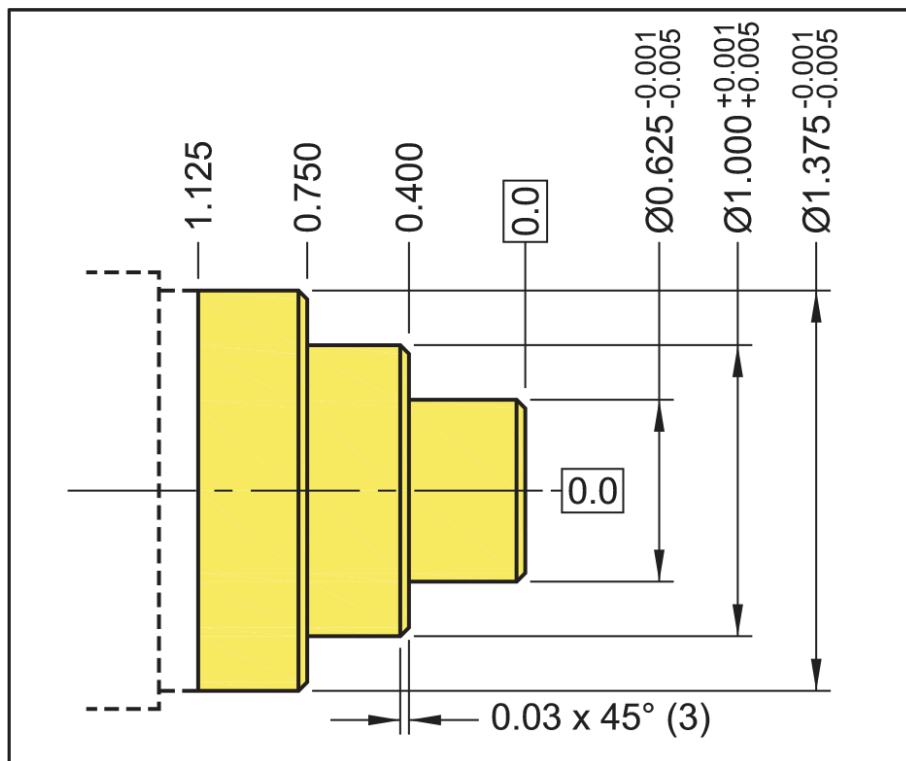


Figure 34-3

Multiple offsets - example for diameters only - O3401

Programming solution is to include *two* offsets for finishing, for example, T0313 and T0314. In the control, correct offset amounts have to be set before machining - the ideal amounts for

middle tolerance are shown:

X-0.003 Z0.000

X+0.003 Z0.000

Note that the Z-offset (it controls shoulder lengths) must be the *same for both wear offsets*.

Here is the complete program - O3401:

O3401

(1.5 ALUMINUM BAR - EXTEND 1.5 FROM JAWS)

(T01 - FACE AND ROUGH TURN)

N1 G20

N2 G50 S3000 T0100

N3 G96 S500 M03

N4 G00 G41 X1.7 Z0 T0101 M08

N5 G01 X-0.07 F0.005

N6 Z0.1

N7 G00 G42 X1.55

N8 G71 U0.1 R0.02

N9 G71 P10 Q17 U0.04 W0.004 F0.01

N10 G00 X0.365

N11 G01 X0.625 Z-0.03 F0.003

N12 Z-0.4

N13 X1.0 C-0.03 (K-0.03)

N14 Z-0.75

N15 X1.375 C-0.03 (K-0.03)

N16 Z-1.255

N17 U0.2

N18 G00 G40 X5.0 Z5.0 T0100 N19 M01

(T03 - FINISH TURN)

N20 G50 S3500 T0300

(-- OFFSET 00 AT THE START OF THE TOOL -----)

N21 G96 S750 M03

N22 G00 G42 X1.7 Z0.1 T0313 M08

(-- OFFSET 13 FOR THE 0.625 DIAMETER -----)

N23 X0.365

N24 G01 X0.625 Z-0.03 F0.002

N25 Z-0.4

N26 X1.0 C-0.03 (K-0.03) T0314

(-- OFFSET 14 FOR THE 1.0 DIAMETER -----)

N27 Z-0.75
 N28 X1.375 C-0.03 (K-0.03) T0313
 (-- OFFSET 13 FOR THE 1.375 DIAMETER -----)
 N29 Z-1.255
 N30 U0.2
 N31 G00 G40 X5.0 Z5.0 T0300
 (-- OFFSET 00 AT THE END OF TOOL -----)
 N32 M01
 (T05 - 0.125 WIDE PART-OFF)
 N33 T0500
 N34 G97 S2000 M03
 N35 G00 X1.7 Z-1.255 T0505 M08
 N36 G01 X1.2 F0.002
 N37 G00 X1.45
 N38 Z-1.1825
 N39 G01 X1.315 Z-1.25 F0.001
 N40 X-0.02 F0.0015
 N41 G00 X5.0
 N42 Z5.0 T0500 M09
 N43 M30
 %

This is the complete program, using all three tools required. Since T01 and T05 do not change for any forthcoming examples, only T03 will be shown from now on.

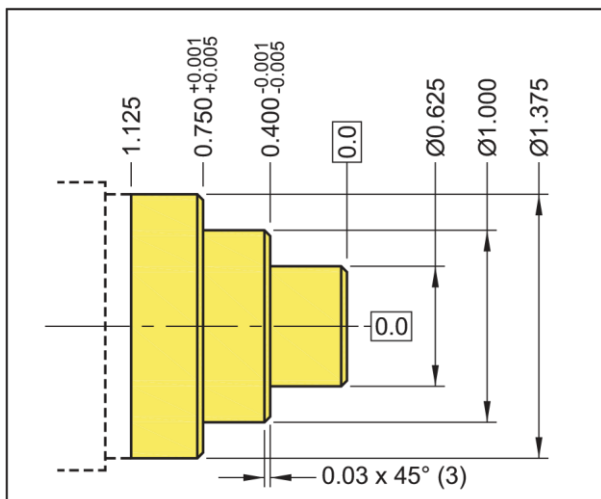


Figure 34-4
Multiple offsets - example for shoulders only - O3402

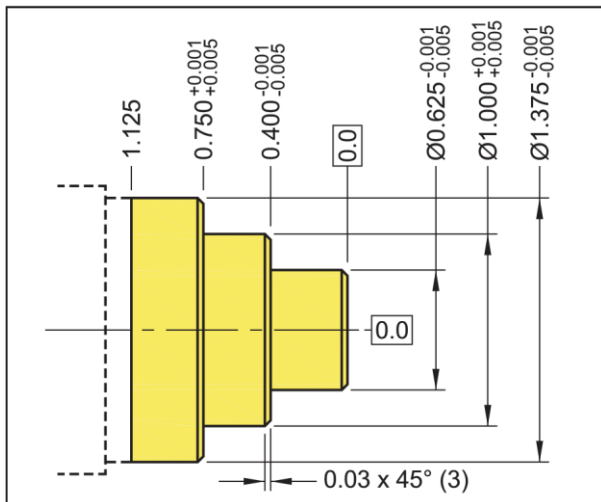


Figure 34-5

Multiple offsets - example for diameters and shoulders - O3403

Shoulder Tolerances

Example drawing shown in Figure 34-4 illustrates the same basic sample part with variable tolerances specified only on the *shoulders*.

Programming solution is to include *two* offsets for finishing cut, for example T0313 and T0314. In the control, their amounts have to be set before machining - the ideal amounts for middle tolerance are shown:

X0.0000 Z+0.0030

X0.0000 Z-0.0030

Note that in this case, the X-offset (it controls size of the diameters) must always be the same for both offsets. Here is T03 for program O3402:

O3402

...

(T03 - FINISH TURN)

N20 G50 S3500 T0300

(-- OFFSET 00 AT THE START OF TOOL -----)

N21 G96 S750 M03

N22 G00 G42 X1.7 Z0.1 T0313 M08

(-- OFFSET 13 FOR THE 0.4 SHOULDER -----)

N23 X0.365

N24 G01 X0.625 Z-0.03 F0.002

N25 Z-0.4

N26 X1.0 C-0.03 (K-0.03)

N27 Z-0.75 T0314

(-- OFFSET 14 FOR THE 0.75 SHOULDER -----)

N28 X1.375 C-0.03 (K-0.03)

N29 Z-1.255

N30 U0.2

N31 G00 G40 X5.0 Z5.0 T0300

(-- OFFSET 00 AT THE END OF TOOL -----)

N32 M01 ...

Diameter and Shoulder Tolerances

Example drawing shown in *Figure 34-5* illustrates the same basic sample part with variable tolerances specified on both the *diameters* and *shoulders*.

Programming solution is to include *four* offsets for finishing, for example T0313, T0314, T0315 and T0316. In the control, their amounts have to be set before machining the ideal amounts for middle tolerance are shown:

X-0.0030 Z+0.0030

X+0.0030 Z+0.0030

X+0.0030 Z-0.0030

X-0.0030 Z-0.0030

This is the most intensive version. Not only it is extremely important *where exactly* the offsets appear in the program, but their actual input amount is also critical.

Note that the four X-offsets (they control size of the diameters) tie up with the four Z-offsets (which control the length of shoulders). Here is T03 for program O3403:

O3403

...

(T03 - FINISH TURN)

N20 G50 S3500 T0300

(-- OFFSET 00 AT THE START OF TOOL -----)

N21 G96 S750 M03

N22 G00 G42 X1.7 Z0.1 T0313 M08

(-- OFFSET 13 FROM Z OVER TO Z UNDER ONLY ---)

N23 X0.365

N24 G01 X0.625 Z-0.03 F0.002

N25 Z-0.4

N26 X1.0 C-0.03 (K-0.03) T0314

(-- OFFSET 14 FROM X UNDER TO X OVER ONLY ---)

N27 Z-0.75 T0315

(-- OFFSET 15 FROM Z UNDER TO Z OVER ONLY ---) N28 X1.375 C-0.03 (K-0.03) T0316

(-- OFFSET 16 FROM X OVER TO X UNDER ONLY ---)

N29 Z-1.255

N30 U0.2

N31 G00 G40 X5.0 Z5.0 T0300

(-- OFFSET 00 AT THE END OF TOOL -----)

N32 M01

...

CNC operator must always be aware of the existence of multiple offsets in a program as well as programmer's reason for using them. Initial settings are always critical, so are all changes during machining. As can be seen in the first two examples, programs O3401 and O3402, one group of offsets must always remain the same (X or Z offsets). For instance, in program O3401, diameters are controlled by offsets 03 and 13. That means the Z-offset setting must be the same - *always!* That also means, if there is a need to shift the shoulders 0.002 to the left, for example, *all shoulders must be shifted by the same amount:*

X-0.0030 Z-0.0020

X+0.0030 Z-0.0020

Failure to do that will result in inaccurate dimensions.

D. OFFSET SETTING

The OFFSET screen selected by pressing a key on the control panel will initially display tool geometry and tool wear offsets. They are identical, except the title at the screen top. A typical display will resemble this screen layout (no offsets set):

OFFSET (GEOMETRY)				
NO.	X-AXIS	Z-AXIS	RADIUS	TIP
01	0.0000	0.0000	0.0000	0
02	0.0000	0.0000	0.0000	0
03	0.0000	0.0000	0.0000	0
...	...	...	...	...

X-axis and Z-axis are often shown just as X and Z, *Radius* is shown as R, and *Tip* is shown as T.

The NO. is the offset number, either the first pair of the T address - for the *Geometry* offset, or the second pair - for the *Wear* offset. X-axis and Z-axis are the columns where actual offset amounts are entered for each number, the *Radius* and *Tip* columns are only used if tool nose radius offset is programmed. In that case, *Radius* will be the tool nose radius and *Tip* will be an arbitrary number, as defined by Fanuc, specifying the tool tip orientation. This subject has been described starting on *page 275*.

E. FUNCTIONS FOR GEAR RANGES

A number of CNC lathes are designed to work in several ranges of gear engagement. This machine feature enables the programmer to coordinate required spindle speed with specific power requirements of the machine. As a general rule, the higher the requirement for spindle speed, the lower the maximum available power rating will be, and vice versa. Ranges of spindle speed and power ratings for each range are determined by the machine manufacturer and must never be changed.

Depending on the CNC lathe size, one, two, three, or four gear ranges may be available. Small lathes, or those designed with ultra-high spindle speeds, may have no programmable gear range at all, which means only a single default gear range is available. Very large lathes may have all four gear ranges - and the maximum available spindle speed is usually low in comparison. The most

common average is two gear ranges.

Miscellaneous functions for gear ranges may vary, but are typically M41, M42, M43 and M44, and assume the definition relative to the number of gear ranges available:

Range	Number of available ranges			
	1	2	3	4
Low	-	M41	M41	M41
Medium Low	-	-	-	M42
Medium	-	-	M42	-
Medium High	-	-	-	M43
High	-	M42	M43	M44

Once a certain gear range is selected, the spindle speed range is limited. If the exact range of spindle speed is important, always make an effort to find out the available spindle speeds in each range. Don't be surprised to find out that on most CNC machines, one rpm (1 r/min) is very rare. Typical lowest spindle speed may be around 20 to 30 r/min. Also, don't be surprised to find that there is an overlap, often quite large, for spindle speeds in two ranges. For example, if the *Gear 1* has a range 20 to 1400 r/min, *Gear 2* may have a range of 750 to 2500 r/min. When using spindle speeds available in either range, such as 1000 r/min, selection of gear range is not critical, but low gear range will produce more power.

Here is an actual, although unrelated, example:

Low gear range: 20 - 1075 r/min (M41)

High gear range: 70 - 3600 r/min (M42)

F. AUTOMATIC CORNER BREAK

In CNC turning and boring, there are occasions where the cut from a shoulder to a diameter (or from a diameter to a shoulder) requires a corner break. Breaking a sharp corner is a common practice when machining between shoulders and diameters. Many engineering drawings specify that all sharp corners are to be broken, often without suggesting their size. It is up to the programmer to decide, usually within the range of 0.005 to 0.020 inches (0.125 to 0.500 mm). The required corner break may be either a *chamfer at a 45° angle*, or a *90 sweep blend radius* - both usually small. If the size of corner break is specified, then it must be applied. Corner breaking has three practical reasons:

Functionality

... for strength, ease of assembly, and clearances

Safety

... sharp corners are dangerous

Appearance

... the finished part looks better

In lathe work, many corner breaks apply to cuts between a shoulder and the adjacent diameter (the cut takes a 90° turn in one axis at a time). The start and end points calculation is not difficult but can be time consuming for some jobs, such as shaft turning with many different diameters.

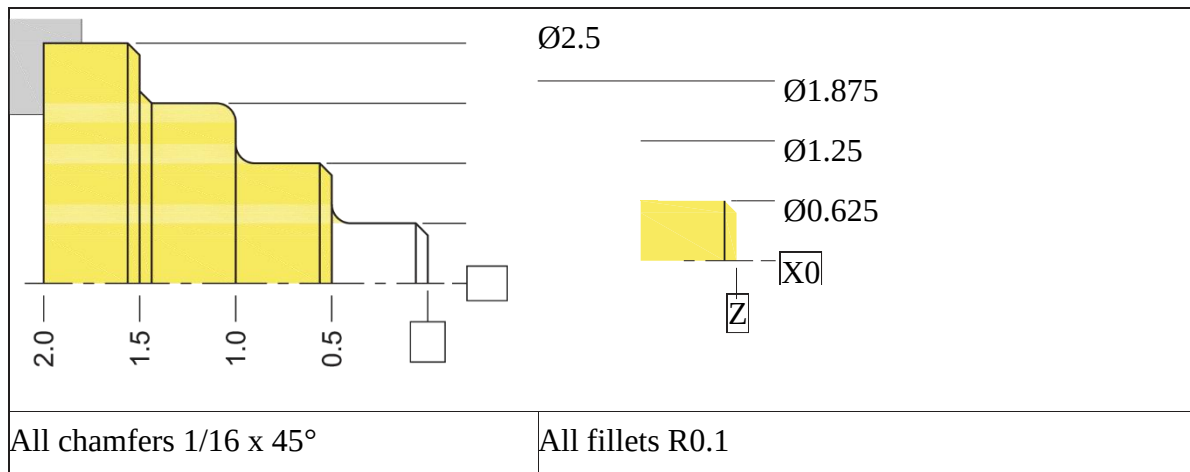


Figure 34-6

Example for an automatic corner break (chamfers and radii)

Drawing in Figure 34-6 shows a simple external part that contains several corners that will benefit from automatic corner break programming feature - not all corners in the drawing qualify (!).

Compare the two methods, to better understand the differences applied in programming. If the programmer *does not* use automatic corner break, each contour change point must be calculated manually, and the result will be rather long program O3404:

O3404 (MANUALLY CALCULATED CORNER BREAK USED) ...

```

N51 T0100
N52 G96 S450 M03
N53 G00 G42 X0.3 Z0.1 T0101 M08
N54 G01 X0.625 Z-0.0625 F0.003
N55 Z-0.4
N56 G02 X0.825 Z-0.5 R0.1
N57 G01 X1.125
N58 X1.25 Z-0.5625
N59 Z-0.9
N60 G02 X1.45 Z-1.0 R0.1
N61 G01 X1.675
N62 G03 X1.875 Z-1.1 R0.1
N63 G01 Z-1.4375
N64 X2.0 Z-1.5
N65 X2.375
N66 X2.55 Z-1.5875
N67 U0.2
N68 G00 G40 X10.0 Z5.0 T0100 N69 M01

```

Only the finished contour is shown (no facing), starting at a *selected* clearance of Z0.1, with *calculated* X-diameter at X0.3. Each contour change point has to be carefully calculated. At the contour end, the last chamfer has been completed at a *selected* clearance of 0.025 above the top

diameter, at X2.55, and *calculated* Z-axis at Z-1.5875.

As always in manual work, the possibility of errors can be significant. For example, one very common error in this type of programming is the target value of X-axis. In turning, it is easy to forget to double the chamfer or radius value (or half it for boring). The result is that block N56 may be:

N56 G02 X0.725 Z-0.5 R0.1 (ERROR IN X)

instead of the *correct* block

N56 G02 X0.825 Z-0.5 R0.1 (X IS CORRECT)

Remember: X = diameter. So, what can be done in the program in order to implement the automatic corner break?

Fanuc control system offers two programming methods that relate to automatic corner breaking on lathes:

Chamfering method... for a 45 chamfer

Blend radius method ... for a 90 blend

Both methods work in a very similar manner and certain rules have to be observed in both cases.

Chamfering at 45 Degrees

Automatic corner chamfering will always take place in the G01 mode, and two special vectors I and K are available for this purpose or a C-vector on some models.

For automatic chamfer generation, the vectors I and K specify *the direction and the amount of cut for the required chamfer*:

'I' VECTOR

is used to create a chamfer starting from the X-axis,

into the X+Z-, X-Z-, X+Z+, or X-Z+ direction

'K' VECTOR

is used to create a chamfer starting from the Z-axis, into the Z-X+, Z-X-, Z+X+, or Z+X- direction

The I and K vector definition is illustrated in *Figure 34-7*.

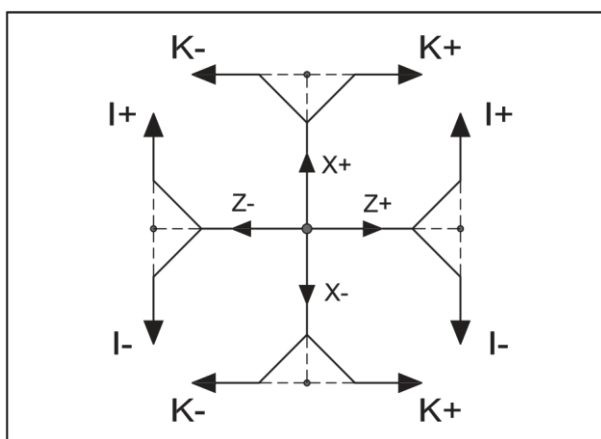


Figure 34-7

Vectors I and K for automatic corner chamfering

When the control system encounters a block containing chamfering vector I or K, it will automatically *shorten* the active programmed tool path length by the amount of the I or K vector, as specified in the program. If not sure whether the I or the K vector should be programmed for

automatic chamfering, consult the above illustration, or apply the following rules:

The vector I indicates the chamfering amount and motion direction when the tool motion is in the order of Diameter-Chamfer-Shoulder, which means tool is cutting along the Z-axis before the chamfer. The chamfer direction can only be from the Z-axis towards the X-axis, with the I vector programmed between:

G01 Z-1.75 I0.125 (CUTTING ALONG Z AXIS)

X4.0 (CONTINUING IN X AXIS AFTER CHAMFER)

Vector K indicates the chamfering amount and motion direction when the tool motion is in the order of Shoulder-Chamfer-Diameter, which means tool is cutting along the X-axis before the chamfer. The chamfer direction can only be from the X-axis towards the Z-axis, with the K vector programmed between:

G01 X2.0 K-0.125 (CUTTING ALONG X AXIS)

Z-3.0 (CONTINUING IN Z AXIS AFTER CHAMFER)

In either case, the sign of I or K vector defines the chamfer cutting direction, within the coordinate system:

Positive value of I or K vector indicates the chamfering direction into the plus direction of the axis *not* specified in the chamfering block

Negative value of I or K vector indicates the chamfering direction into the minus direction of the axis *not* specified in the chamfering block

Values of I and K vectors are always single values (*i.e.*, radius values, *not* diameter values).

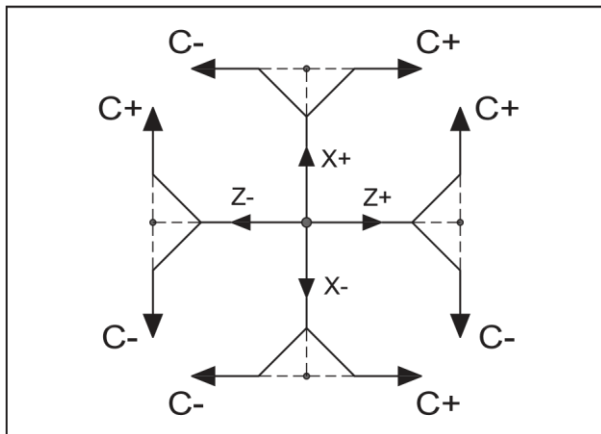


Figure 34-8

Vectors C for automatic corner chamfering

Many latest controls use vectors C+ and C- that replace the I+, I-, K+ and K- vectors - *Figure 34-8*. This is a much simpler programming method, and its applications are the same as for the blend radius R, described shortly. There is no distinction between axes vector selection, just the specified direction:

The C vector is used

... to create a chamfer starting from the X-axis, into the X+Z-, X-Z-, X+Z+, or X-Z+ direction

- or -

... to create a chamfer starting from the Z-axis, into the Z-X+, Z-X-, Z+X+, or Z+X- direction

If the control unit allows C+ or C- vectors, the programming is much easier, as long as the

motion direction is watched carefully. The two previous examples will be:

G01 Z-1.75 C0.125 (CUTTING ALONG Z AXIS)

X4.0 (CONTINUING IN X AXIS AFTER CHAMFER)

G01 X2.0 C-0.125 (CUTTING ALONG X AXIS)

Z-3.0 (CONTINUING IN Z AXIS AFTER CHAMFER)

As was the case with I and K vectors, the C vector is also specified as a single value *per side*, not per diameter.

Blend Radius at 90 Degrees

A blend 90° sweep radius between a shoulder and diameter (or vice versa) is programmed in a similar way as the automatic 45° chamfer using the C vector. *It also takes place exclusively in the G01 mode!* Only one special vector R is used. For automatic blend radius, the vector specifies the *direction and amount of cut for the radius*:

The R vector is used

... to create a blend radius starting from the X axis, into the X+Z-, X-Z-, X+Z+, or X-Z+ direction

- or -

... to create a blend radius starting from the Z axis, into the Z-X+, Z-X-, Z+X+, or Z+X- direction

The R vector definition is illustrated in *Figure 34-9*.

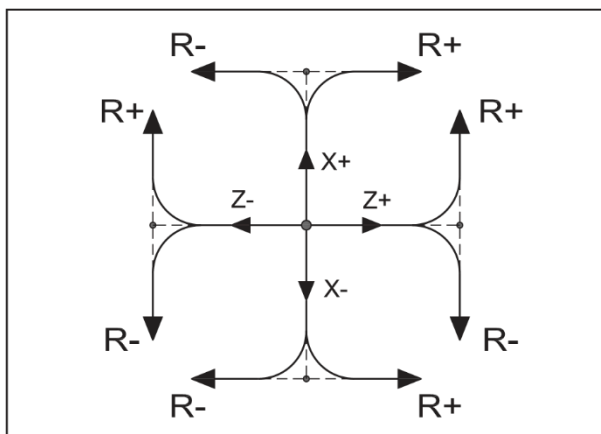


Figure 34-9

Vector R for automatic corner rounding (blend radius)

When the control system encounters a program block containing blend radius vector R, it will automatically *shorten* the current programmed tool path length by the R vector amount, as specified in the program. If not sure whether the R vector should be programmed for automatic blend radius, consult the above illustration or apply the following rule:

Vector R indicates the *radius amount and motion direction* when the tool motion is in the order of *Shoulder Radius-Diameter*, which means tool is cutting along the X-axis before the radius. The same vector is also used when the *radius amount and motion direction* is in the opposite order of *Diameter-Radius-Shoulder*, which means cutting along the Z-axis before the radius.

Radius deviation can be from the X-axis towards the Z-axis, when the R vector is programmed:

G01 X2.0 R-0.125 (CUTTING ALONG X-AXIS)

Z-3.0 (CONTINUING IN Z AXIS AFTER RADIUS)

Radius direction can also be from the Z-axis towards the X-axis, when the R vector is programmed:

G01 Z-1.75 R0.125 (CUTTING ALONG Z AXIS)

X4.0 (CONTINUING IN X AXIS AFTER RADIUS)

In either case, the sign of R vector defines the radius cutting direction, within the coordinate system:

Positive value of R vector indicates the radius direction into the *plus* direction of the axis *not* specified in the radius block

Negative value of R vector indicates the radius direction into the *minus* direction of the axis *not* specified in the radius block

Programming Conditions

Breaking corners automatically makes programming for modern CNC lathes a lot easier, as only drawing dimensions are used and no external manual calculations are necessary. Regardless of whether program contains vectors I or K or C for chamfering, or vector R for blend radius corner, the basic conditions and general rules are very similar:

Chamfer or radius must be fully contained in a single quadrant - 90 only

Chamfers must have a 45 angle and radii must have a 90 sweep angle between a shoulder and diameter or a diameter and shoulder

Amounts of chamfering vectors I and K or C, as well as radius vector R, are always single values meaning *per side* values, not diameter values

Direction of cut before corner rounding must be perpendicular to the direction of cut after rounding, along one axis only

Direction of cut following the chamfer or radius must continue along a single axis only, and must have the length equivalent to at least the chamfer length or the radius amount - the cutting direction cannot reverse

Both chamfering and blend radius corner breaking takes place in G01 mode (linear interpolation mode)

When writing CNC program, only the known intersection from the drawing - *the sharp point* - is required. That is the point between shoulder and diameter, without the chamfer or radius being considered

These rules apply equally to turning and boring CNC lathe operations. Study them carefully to avoid problems at the machine later.

Programming Example

Following program (O3405) combines both chamfering and blend radius vectors into a complete example. The same drawing is used for this version, as for the traditional method, illustrated earlier in *Figure 34-6*.

In order to fully appreciate all differences between the two programming methods (both are technically correct), compare the following program O3405 with the earlier program O3404. I and K vectors are used for chamfering, as they are more difficult than the C vectors:

O3405 (AUTOMATIC CORNER BREAKS USED) ...

N51 T0100

N52 G96 S450 M03

N53 G00 G42 X0.3 Z0.1 T0101 M08

N54 G01 X0.625 Z-0.0625 F0.003

N55 Z-0.5 R0.1

N56 X1.25 K-0.0625

N57 Z-1.0 R0.1

N58 X1.875 R-0.1

N59 Z-1.5 I0.0625

N60 X2.375

N61 X2.55 Z-1.5875

N62 U0.2

N63 G00 G40 X10.0 Z5.0 T0100 N64 M01

Although this program is shorter, the five blocks saved in the program offer only a small benefit. The real benefits mean no G02s, no G03s, no start/end points of chamfers or radiuses, no center point calculations.

Except for the contour beginning and end, this type of programming greatly enhances program development and allows for very fast and easy changes during machining, if necessary. If a chamfer or a blend radius is changed in the drawing, only a *single value* has to be changed in the program, without any recalculations. Of course, the rules and conditions mentioned earlier must be always observed. The main benefit of automatic corner breaking are the ease of changes and the absence of manual calculations.

G. ROUGH AND FINISHED SHAPE

The vast majority of material removal on CNC lathe is done by using various cycles, described in detail in the next chapter. These cycles require input of data that is based on machining knowledge, such as depth of cut, stock allowance, speeds and feeds, etc.

Rough and finished shapes often require manual calculations, using algebra and trigonometry. These calculations should be done on separate sheets of paper, rather than in the drawing itself. That way, the work is better organized. Also, if there is a change later, for example, an engineering design change, it is easier to keep track of what is where.

Rough Operations

A great part of lathe machining amounts to removal of excessive stock to create a part, almost completed. This kind of machining is generally known as *roughing*, rough turning, or rough boring. As a machining operation, roughing does not produce a high precision part, that is not the purpose of roughing. Its main purpose is to remove unwanted stock efficiently, which means fast and with maximum tool life, and leave suitable all-around stock for finishing. Cutting tools used for roughing are strong, usually with a relatively large nose radius. These tools have to be able to sustain heavy depths of cut and high cutting feeds. Common diamond shaped tools suitable for roughing are

80° inserts (up to 2+2 cutting corners), and trigon inserts (up to 3+3 cutting corners). 2+2 or 3+3 means on 2 or 3 *cutting edges on each side* of the insert. Not all inserts can be used from both sides. *Figure 34-10* shows some typical tools and orientation for rough turning and boring.

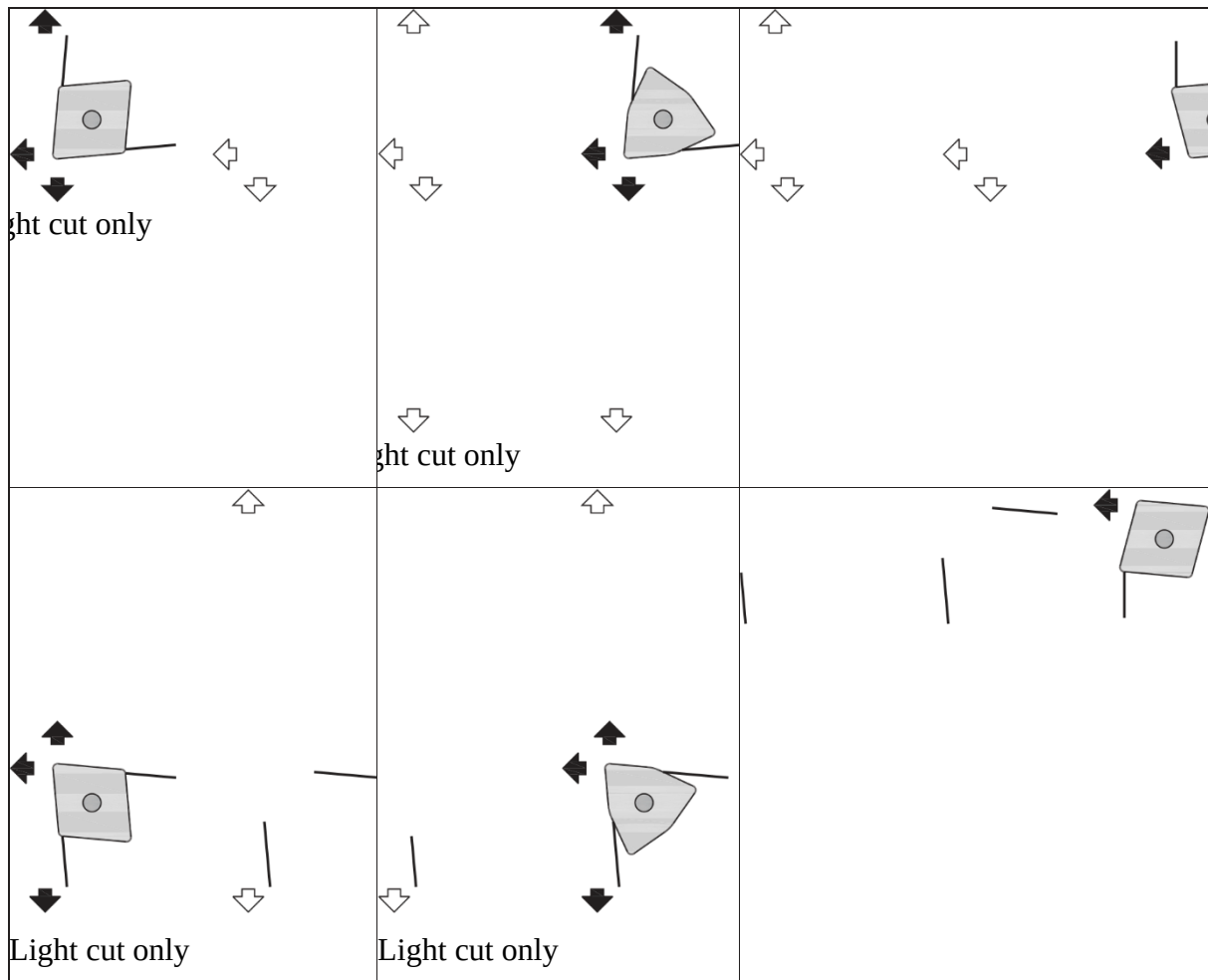


Figure 34-10

Tool orientation and cutting direction for roughing.

Upper row shows external tools, lower row shows internal tools.

Although a number of tools can be programmed in several directions, some directions are not recommended at all, or only for light or medium light cuts.

In practice, always follow one basic rule of machining this rule is valid for all types of machines:

Always do heavy operations before light operations

This basic rule means that *all* roughing should be done before the first finishing cut is programmed. The reason here is to prevent a possible shift of the material during roughing, after some finishing had already been done.

For example, the requirement is too rough and finish both external and internal diameters. If the above rule is applied to these operations, roughing outside of the part will be first, then roughing inside of the part, and only then applying the finishing cuts. It really does not matter whether the rough machining is done first externally or internally, as long as it gets done before any finish cuts, which also can be in either order.

Tool wear can be minimized if the depth of cut is sufficient and the cutting radius gets 'under the skin' of the material, usually during the first cut. Coolant is usually a must for most materials and should be applied *before* the tool actually contacts the part.

Finish Operations

Finish operations take place as the final cutting motions, after most of the stock has been removed (roughed out), leaving only a small amount of overall stock for finishing. Finishing cutting tool can have smaller nose radius and, for even a better surface finish, higher spindle speeds

and lower cutting feeds are typical.

Many different tools can be used for finishing operations as well, but the most typical finishing tools are two diamond shaped inserts, with a 55 and 35 insert angle. Their shape, common orientation and cutting directions are shown in *Figure 34-11*.

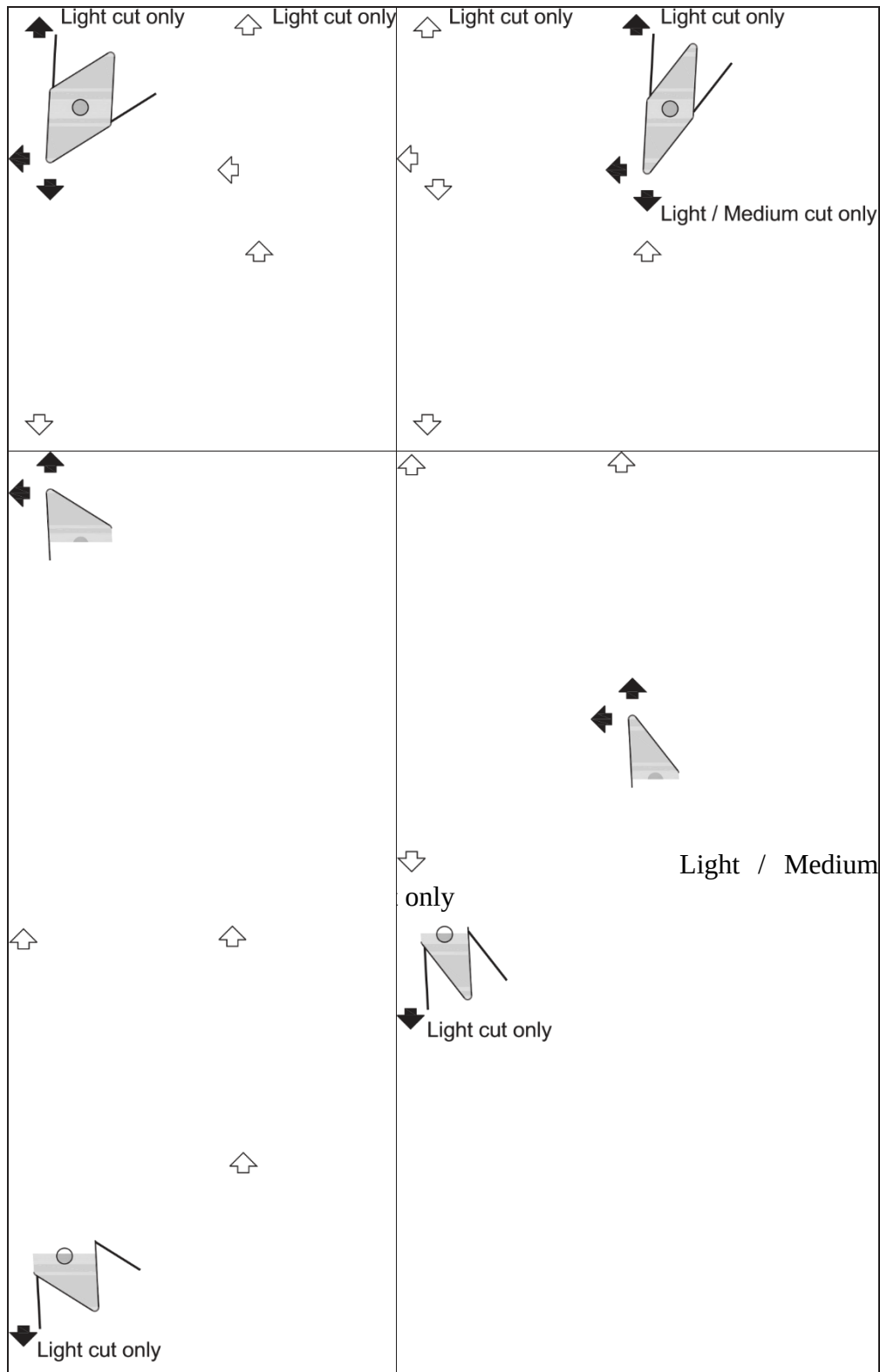


Figure 34-11

Tool orientation and cutting direction for finishing with common lathe tools. Upper row shows external tools, lower row shows internal tools.

Note that some cutting directions are only recommended for light or medium cuts. Why? The

answer has a lot to do with the amount of material (stock) the tool removes in the specified direction.

Stock and Stock Allowance

Material used for machining is often called *stock*. When the tool removes stock to cut a desired shape, it can only handle a certain amount of it at a time. Insert shape, its orientation toward the part and its cutting direction, the insert size and thickness, all have a profound effect on the allowable stock to be removed. This is especially important in semi finishing and finishing operations. Stock allowance specifies the amount of material left for these fine operations. If too much material or too little material is left to be cut during finishing, part accuracy and surface finish quality will suffer. Also, carefully consider not just a stock allowance overall on the part, but *individual stock allowances* for the X and Z axes!

As before, there is a general rule of thumb, that on the X-axis, that is for cutting diameters, leave the stock *equivalent to* or *slightly larger* than radius of the subsequent finishing tool. For example, if a 0.031 inch tool nose radius (0.80 mm) is used for finishing, leave 0.030 to 0.040 inch stock (about 1 mm). That is the physical stock, the actual stock amount assigned *per side*, not on diameter!

The amount of stock left on Z-axis (typically for facing shoulders at 90°) is much more critical. If cutting along the positive X-axis only (for turning), or the negative X-axis only (for boring), with a tool that has a lead angle of 3° to 5, do not leave more than 0.003 to 0.006 inch (0.080 to 0.150 mm) on any straight shoulder. *Figure 34-12* shows the effect of too much stock allowance for certain cutting directions and a method to eliminate it.

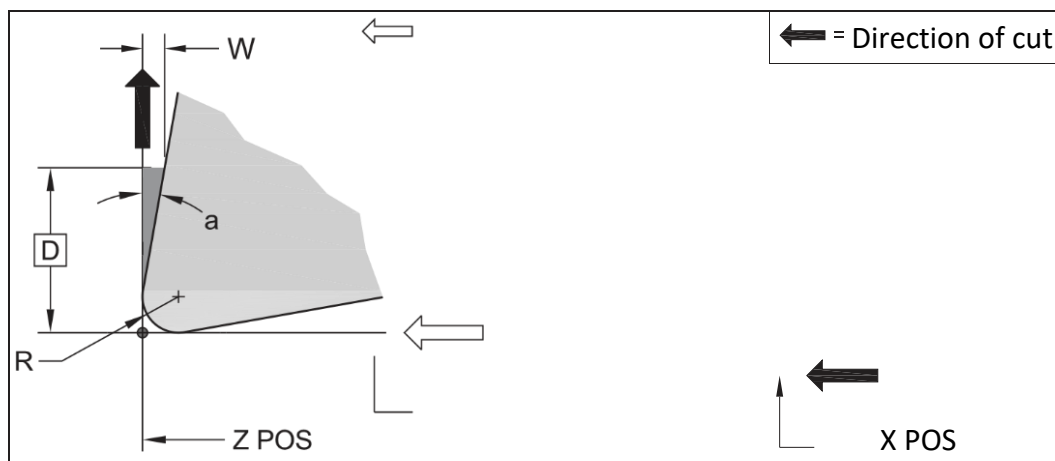


Figure 34-12

Effect of stock allowance W on depth of cut D

In the illustration, actual depth of cut D at the face Z POS, is determined by the amount of stock W. To calculate the depth D, use this formula:

$$D = \frac{R}{\tan A} - \frac{W}{\tan A}$$

+ where ...

D = Actual depth of cut at the face A = Lead angle of the insert

R = Radius of the insert

W = Stock left on face for finishing X POS = Target position for the X-axis
axis Z POS = Target position for the Z-axis

Illustration applies equally to boring, when the X-axis direction is opposite the one shown. To understand better the consequences of a heavy stock left on the face, evaluate this example:

Example:

Amount of stock left on face is 0.030, tool nose radius is 0.031, and tool lead angle is 3:

$$W = 0.030, R = 0.031, A = 3$$

There is enough data available to calculate the unknown depth D, using the above formula:

$$D = \tan 3/2 \cdot 0.031 + 0.030 / \tan 3 + 0.031 \quad D = 0.60425$$

For an insert with a 0.500 inch inscribed circle (such as DNMG-432, for example), the actual depth of cut at the face will be 0.60425 - *more than any reasonable amount!*

Since the earlier suggestion was no more than 0.006, recalculate the example for the largest depth, if the $W=0.006$:

$$D = \tan 3/2 \cdot 0.031 + 0.006/\tan 3 + 0.031 \quad D = 0.14630$$

That is a more reasonable depth of cut at the face, so the Z axis stock allowance of 0.006 can be used. For facing in the opposite X direction or for not unidirectional faces, leave stock much bigger, usually close to the tool radius.

H. PROGRAMMING A RECESS

Another very important aspect of programming for CNC lathes is the change of cutting direction. Normally, program a tool motion in such a way that the motion direction from the starting point will be:

Positive X direction for external machining ... and / or ...

Negative Z direction for external machining

Negative X direction for internal machining

... and / or ...

Negative Z direction for internal machining

There are also *back turning*, or *back boring* operations used in CNC programming, but these are just related and less common variations of the common machining. In the most common machining on CNC lathes, any change of direction in a single axis into the material constitutes an undercut, a cavity, or more commonly known - a *recess*.

A recess is commonly designed by the engineers to relieve - or undercut - a certain portion of the part, for example, to allow a matching part to fit against a shoulder, face, or surface of the machined part.

In CNC lathe programming, a recess can be machined very successfully with any tool that is used with the proper depth of cut, and a *suitable back angle clearance*. It is the second requirement that will be looked at next.

Figure 34-13 shows a simple drawing of a roller. In the middle of the object, there is an undercut (recess) between the 1.029 and the 0.939. The objective is to calculate, *not to guess*, what is the maximum back angle tool that can be used for cutting this recess in a single operation.

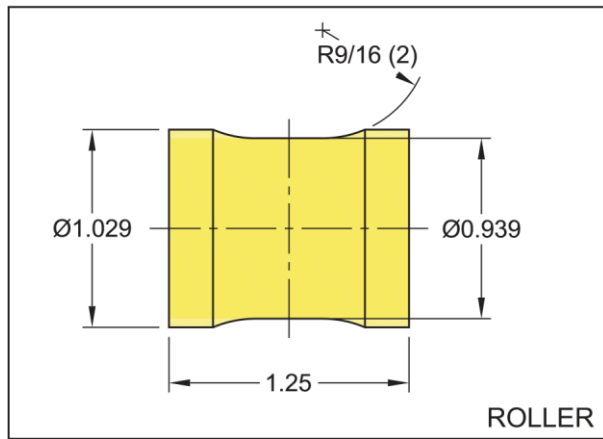


Figure 34-13

Back angle clearance calculation example

The first step is to consider the drawing - that is always the given and unchangeable source of data. Calculated difference between the diameters, and the recess radius will be required. Figure 34-14 illustrates the generic details of the *provided data* (except the angle b) from the drawing.

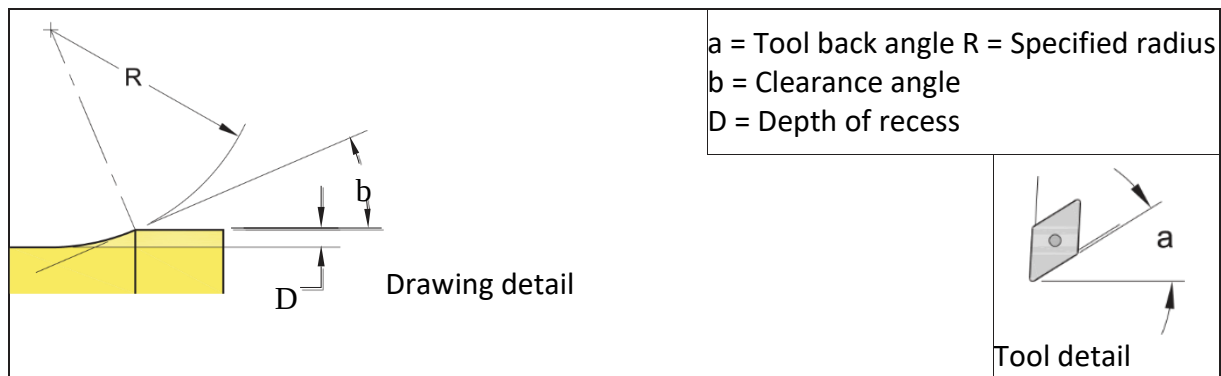


Figure 34-14

Data required to calculate angle 'b'

Formula required to calculate the angle b uses simple trigonometric formula. First, calculate the depth of the recess D , which is nothing more than one half of the difference between the two given diameters:

$$D = \frac{\text{LARGE DIA} - \text{SMALL DIA}}{2}$$

Once the recess depth D is known, formula to calculate the angle b can be used:

$$b = \cos^{-1} \left(\frac{R - D}{R} \right)$$

For the example, calculation will be:

$$b = \cos^{-1} \left(\frac{0.5625 - 0.045}{0.5625} \right) = 23.07392$$

For actual machining, select a tool with the back angle a greater than the calculated angle b . For the illustrated drawing (23.07 required clearance), the selected tool could be either a 55° diamond shape (back angle clearance a is 30 to 32), or a 35-diamond shape (back angle clearance a is 50 to 52) - both are greater than the calculated minimum clearance. The actual angles depend on the tool manufacturer, so a tooling catalogue is a good source of data.

This type of calculation is important for any recesses, undercuts and special clearances, whether programmed with the help of cycles or developed block by block. The example only illustrates one possibility but can be used for any calculations where the back angle clearance is required.

I. SPINDLE SPEED IN CSS MODE

From several earlier topics, remember that the abbreviation CSS stands for *Constant Surface Speed*. This CNC lathe feature will constantly keep recalculating the actual spindle speed in *revolutions per minute* (r/min), based on the programmed input of surface speed. The surface speed is programmed in *feet per minute* - ft/min (imperial system) or in *meters per minute* - m/min (metric system).

In the program, the '*per minute*' input uses preparatory command G96, as opposed to the direct *r/min* input using command G97.

Constant Surface Speed is a powerful feature of the control system and without it, we would look back many years. There is a rather small problem associated with this feature, often neglected altogether, or at least not considered important enough. This rather 'small problem' will be illustrated in a simple program example.

This program example covers only a few blocks at the beginning, when the cutting tool approaches the part. That is enough data to consider the question that follows.

O3406

N1 G20 T0100

N2 G96 S450 M03

N3 G00 G41 X0.7 Z0 T0101 M08 N4 ...

The question is this: What is the actual spindle speed (in *r/min*), when block N2 is executed? Of course, the spindle speed is unknown at the moment. It cannot be known, unless the current *diameter*, the diameter where the tool is located at that moment, is also known. Control system keeps track of the current tool position at all times. So, when block N2 is executed, the actual spindle *r/min* will be calculated for the current diameter, *as stored in the control*, specified in the geometry offset entry. For the example, consider that the current diameter is 23.5 or X23.5.

From the standard *r/min* formula, the spindle speed calculated for 450 ft/min and 23.5 as 73 *r/min* is rather slow but correct. At the next block, block N3, the tool position is rather close to the part, at diameter of 0.700 (X0.7). From the same standard formula, the spindle speed can be calculated for that diameter as 2455 *r/min* - considerably fast but also correct. The problem? There may not be one for every machine, but if ever there is a problem, the following solution will eliminate it.

The possible problem can be linked to the rapid motion from 23.5 to 0.700. The actual travel distance (per side of part) is $(23.5-0.700)/2$, which is 11.400. During rapid travel motion, the cutting tool has to move 11.400 inches and - *at the same time* - change the spindle speed from a slow 73 *r/min*, to a fast 2455 *r/min*. Depending on the control system and its handling of such a situation, the tool may actually start cutting *at a slower spindle speed than was originally intended*.

If such a situation does happen and presents a problem, the only step that can be done is to *preprogram* the expected spindle speed in *r/min*, *before* the cutting tool approach motion, *then* switch to the constant surface speed (CSS) mode and continue:

O3407

N1 G20 T0100

N2 G97 S2455 M03 (R/MIN PRESET)

N3 G00 G41 X0.7 Z0 T0101 M08

N4 G96 S450 M03 N5 ...

What had been done requires more evaluation. What had been done is that the spindle was started at the final expected *r/min*, before the tool reaches the part, in block N2. In block N3, tool moves to the start of cut, while the spindle is already at the peak of programmed speed. Once the target position along the X-axis has been reached (block N3), the corresponding CSS mode can be in effect for all subsequent cuts.

This is an example that does not necessarily reflect everyday programming of CNC lathes. In this situation, some additional calculations have to be done, but if they solve the problem - they are worth the extra effort! Some CAD/CAM system can be set to do exactly that, automatically. If the current X position of the tool is unknown, estimate it.

J. LATHE PROGRAM FORMAT

In review of the already presented examples, a certain consistency can be seen in the program output. This may be called a style, a format, a form, a template, structure, as well as several other terms. Each programmer develops his or her own style over a period of time. A consistent style is important for efficient program development, program changes and program interpretation.

Program Format - Templates

Most examples have followed a certain program format. Note that each CNC lathe program begins with the G20 or G21 command and perhaps some cancellation codes. The block that follows is a tool selection, next is spindle speed data, etc. This format will not basically change from one job to another - it follows a certain consistent pattern which forms the basic *template* for writing the program.

General Program Format

To view the format often enough will forge a mental image in the programmer's mind. The details that are not understood yet will become much clearer after acquiring some general understanding of the relationships and details used in various programming methods. Here is a suggested template (structure) for a CNC lathe program.

General Program Pattern - Lathe:

O.. (PROGRAM NAME)

N1 G20 G40 G99 (PROGRAM START UP)

N2 T..00 M4..(TOOL AND GEAR RANGE)

N3 G97 S.. M03 (STABILIZE R/MIN)

N4 G00 [G41/G42] X.. Z.. T.. M08 (APPROACH)

N5 G96 S.. (CUTTING SPEED)

N6 G01 [X../Z..] F.. (FIRST CUTTING MOTION)

N7 ...

...

... (MACHINING)

...

N.. G00 [G40] X.. Z.. T..00(TOOL CHG POSITION) N.. M01 (OPTIONAL STOP)

...

N.. M30 (PROGRAM END)

%

This generic structure is good for most lathe programs. Feel free to adjust it as necessary. For example, not every job requires spindle speed stabilization, so block N3 will not be necessary. It also means that M03 rotation has to be moved to block N5. Take the general program structure as an example only, not as a fixed format.

Approach to the Part

An important part of any lathe program structure is the method of approaching a revolving part. If the part is concentric, the approach can be similar to the *A* option in *Figure 34-15*. Although a facing cut is illustrated, the approach would be logically the same for a turning or a boring cut. Keep the starting point *SP* well above the diameter, at least 0.100 (2.5 mm) *per side* and more, if the actual diameter is not known *exactly*. The *B* option of tool approach is two single axes at a time. It is a variation of the first example, and the X-axis motion can be further split into a rapid and cutting motion, if required. Finally, the *C* option uses clearance in the Z-axis, far from the front face. Again, final motion toward the face can be split into separate rapid and linear motions.

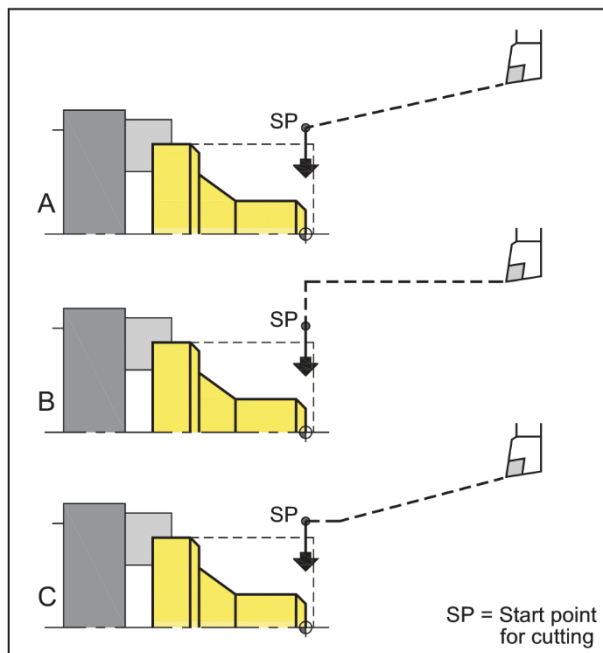


Figure 34-15

Safe approach to a part - example for a facing cut shown

There are many variations on these methods, too numerous to list. The main objective of considering approach towards the part in the first place is safety. A collision of a tool with a revolving part can have serious consequences.

Turning and boring is a large subject. Many other examples could have been included in this chapter. Other chapters in this book also cover turning and boring, but in a more specialized way, for example, turning and boring cycles. Examples that were presented in this chapter should be useful to any CNC lathe programming.

27. LATHE CYCLES

In the last chapter, several lathe procedures described programming of a turning and boring tool path. A number of different techniques have been introduced, mainly describing the *finishing* tool path. Virtually no attention has yet been given to the removal of an excessive stock, in such operations as rough turning and rough boring. It is a subject in its own right and this chapter describes various methods of stock removal for roughing and finishing.

A. STOCK REMOVAL ON LATHES

One of the most time-consuming tasks in manual programming for a CNC lathe is the removal of an excessive stock, typically from a cylindrical material, known as rough turning or rough boring - or simply *roughing*.

To manually program a roughing tool path requires a series of coordinated rough passes, with one block of program for each tool motion. For roughing of a complex contour, such a method is extremely time consuming and very inefficient, as well as prone to errors. Some programmers try to sacrifice programming quality for speed, by leaving an *uneven* stock for finishing, causing the cutting tool to wear out prematurely. Surface roughness of the finished profile often suffers as well.

It is in the area of rough stock removal where the modern lathe controls are very useful and convenient. Almost all CNC lathe systems have a feature that allows the roughing tool path to be processed automatically, using *special cycles*. Roughing is not the only application for these cycles, there are also special cycles available for *threading* and *simple grooving*. The grooving and threading cycles are outside of this chapter but will be covered in detail in the next three chapters.

Simple Cycles

Fanuc and similar controls support a number of special lathe cycles. There are three rather simple cycles that have been part of Fanuc controls for quite a while. They first appeared with the early CNC units and were limited by the technological progress of the time. Various manuals and textbooks refer to them as the *Fixed Cycles* or *Simple Cycles* or even *Canned Cycles*, similar in nature to their cousins for drilling operations on CNC mills and machining centers. Two of these early cycles are used for turning and boring, the third cycle is a very simple threading cycle. This chapter covers the first two cycles.

Complex Cycles

With the advancement of computer technology, control manufacturers have developed cutting cycles capable of very complex lathe operations and made them an integral part of the lathe control systems. These special cycles are called by Fanuc the *Multiple Repetitive Cycles*. Their major improvement over simple cycles are in their excellent flexibility. Some of these advanced cycles cover turning and boring, others grooving and threading.

Don't get misled by the description '*complex*' - these cycles are only complex in the mathematical sense and even then, only internally. They are complex within the control system only. In fact, these very advanced machining cycles are much easier to program than their simple predecessors. In addition, they can also be very easily changed at the machine control, to optimize them for best performance, right on the job.

B. PRINCIPLES OF LATHE CYCLES

Similar to drilling operations for CNC machining centers, all cycles for lathes are based on the same technological principles. The programmer only enters the overall data (typically variable cutting parameters), and the CNC system will calculate the details of individual cuts. These calculations are based on the combination of the fixed and variable data. Return tool motions in all these cycles are automatic, and only the values to be changed are specified within the cycle call.

Simple cycles are designed exclusively to cut a straight cut, with no chamfers, tapers or radii and also with no undercuts. The simple cycles can only be used to cut vertically, horizontally, or at an angle, for taper cutting. These original cycles cannot do the same cutting operations as the more modern and advanced multiple repetitive cycles - for example, they cannot rough out a radius or change direction of the cutting. Simply, they cannot contour.

In the category of simple turning cycles, there are two that do allow removal of rough stock from a cylindrical or conical part. Each block of these cycles replaces four *regular* blocks of the part program. In the category of multiple repetitive cycles, there are several cycles designed for complex roughing, one for finishing, as well as cycles for grooving and threading. Multiple repetitive cycles are capable of some very complex contouring.

C. G90 - STRAIGHT CUTTING CYCLE

Before going further, a reminder. Do not confuse G90 for lathes with G90 for machining centers. In turning, G90 is a lathe cycle, G90 is the absolute mode in milling:

G90 is absolute mode for milling,

X and Z axes are absolute mode for turning

G91 is incremental mode for milling,

U and W axes are incremental mode for turning

A cycle identified by G90 preparatory command (*Type A* group of G-codes) is called the *Straight Cutting Cycle (Box cycle)*. Its purpose is to remove excessive stock between the cutter start position and the coordinates specified by the X and the Z axes. Resulting cut is a straight turning or boring cut, *normally parallel* to the spindle centerline, and the Z-axis is the main cutting axis. As the cycle name suggests, G90 cycle is used primarily for removing stock in a rectangular fashion (box shape). The G90 cycle can also be used for a taper cutting. In *Figure 35-1*, the cycle structure and motions are illustrated.

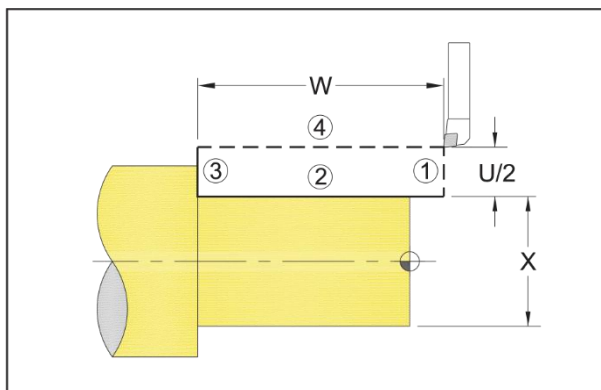


Figure 35-1

G90 simple cycle structure - straight cutting application

Cycle Format

G90 cutting cycle has two predetermined programming formats. The first one is for straight cutting only, along the Z-axis, as illustrated in *Figure 35-1*.

Format 1:

G90 X (U).. Z (W).. F..

+ where ...

X = Diameter to be cut

Z = End of cut in Z position

F = Cutting feed rate (usually *in/rev* or *mm/rev*)

The second format adds the parameter I or R to the block and is designed for taper cutting motions, with the dominance of the Z-axis - *Figure 35-2*.

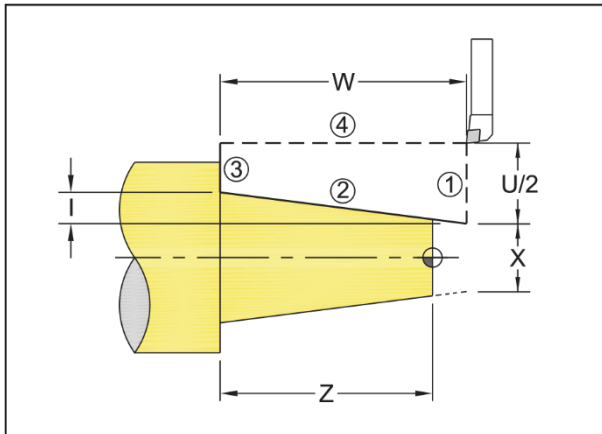


Figure 35-2

G90 cycle structure - taper cutting application Format 2 (two versions):

G90 X (U).. Z (W).. I.. F..

G90 X (U).. Z (W).. R.. F..

+ where ...

X = Diameter to be cut

Z = End of cut in Z position

I (R) = Distance and direction of the taper

(I=0 or R=0 for straight cutting)

F = Cutting feed rate (usually *in/rev* or *mm/rev*)

In both examples, the designation of axes as X and Z is used for absolute programming, indicating the tool position from program zero. The designation of axes as U and W is used for incremental programming, indicating actual travel distance of the tool from the current position. The F address is cutting feed rate, normally in *inches per revolution* or *millimeters per revolution*. The I address is used for taper cutting along the horizontal direction. It has an amount equivalent to *one half* of the distance from the diameter at taper end, to the diameter at taper beginning. The R address replaces the I address and is available on *newer controls only* - its purpose is the same.

To cancel G90 cycle, all that is necessary to do is to use any motion command - G00, G01, G02 or G03. Commonly, it will be G00 rapid motion command:

G90 X (U).. Z (W).. I.. F..

...

...

G00 ...

Straight Turning Example

To illustrate a practical application of G90 cycle, study *Figure 35-3*. It shows rather a simple diameter turning, from a 4.125-inch stock down to a final 2.22 inch, over the length of 2.56 inches. There are no chamfers, no tapers, and no radii. This fact restricts the practical usefulness of the G90 cycle to a very simple roughing only, but still beats the manual alternative.

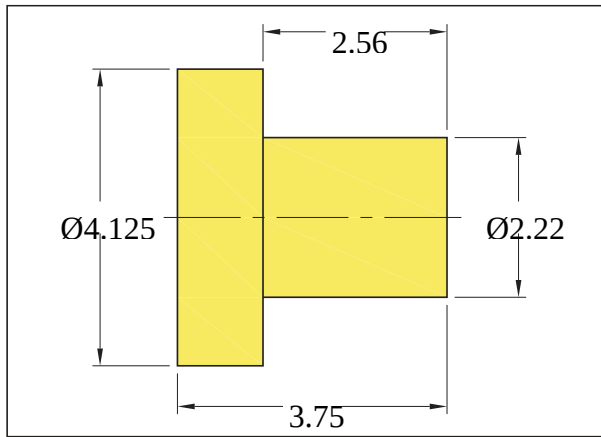


Figure 35-3

Example of G90 cycle in straight cutting - programs O3501 & O3502

Since G90 is a roughing cycle, the *depth* of each cut has to be selected first, then the *stock* amount left for finishing. To decide on the depth of each cut, find out how much stock is actually there to be removed from the diameter. The actual amount of stock is calculated *per side*, as a radius value, along the X-axis:

$$(4.125 - 2.22) / 2 = 0.9525$$

For a 0.030 stock per side for the finishing cut, the 0.030 will be subtracted from the total X-stock, so the total depth amount to remove will be 0.9225. Next is the selection of cut segmentation for the total depth. For five even cuts, each depth of cut will be 0.1845, for six cuts, 0.1538. Six cuts will be selected and 0.030 left per side, or 0.06 on the diameter - the first diameter will be X3.8175. Also, 0.005 stock allowance will be left on the face, so the Z-axis end of cut will be at Z-2.555. The actual clearance above the diameter and in front of the part will be the usual 0.100.

O3501

(G90 STRAIGHT TURNING CYCLE - ABSOLUTE)

N1 G20

N2 T0100 M41

N3 G96 S450 M03

**N4 G00 X4.325 Z0.1 T0101(START
J8 INT)**

N5 G90 X3.8175 Z-2.555 F0.01 (PASS 1)

N6 X3.51 (PASS 2)

N7 X3.2025 (PASS 3)

N8 X2.895 (PASS 4)

N9 X2.5875 (PASS 5)

N10 X2.28 (PASS 6)

N11 G00 X10.0 Z2.0 T0100 M09

N12 M01 (END OF ROUGHING)

If preferred, use the incremental programming method. However, it is easier to trace the program progress with absolute coordinates than incremental distances. However, here is the incremental version:

O3502

(G90 STRAIGHT TURNING CYCLE - INCREMENTAL)

N1 G20

N2 T0100 M41

N3 G96 S450 M03

N4 G00 X4.325 Z0.1 T0101(START
INT)

N5 G90 U-0.5075 W-2.655(PASS 1)
.01

N6 U-0.3075 (PASS 2)

N7 U-0.3075 (PASS 3)

N8 U-0.3075 (PASS 4)

N9 U-0.3075 (PASS 5)

N10 U-0.3075 (PASS 6)

N11 G00 X10.0 Z2.0 T0100 M09

N12 M01 (END OF ROUGHING)

This cycle is quite simple in both versions - all that is needed is to calculate the new diameter for each roughing cut. If the same roughing tool path had been programmed using the block-by-block method (without G90), the final program would be more than three times longer.

Taper Cutting Example

The Figure 35-4 shows a drawing similar to that used for the previous example. In this example, a taper will be cut, also using G90 simple cycle.

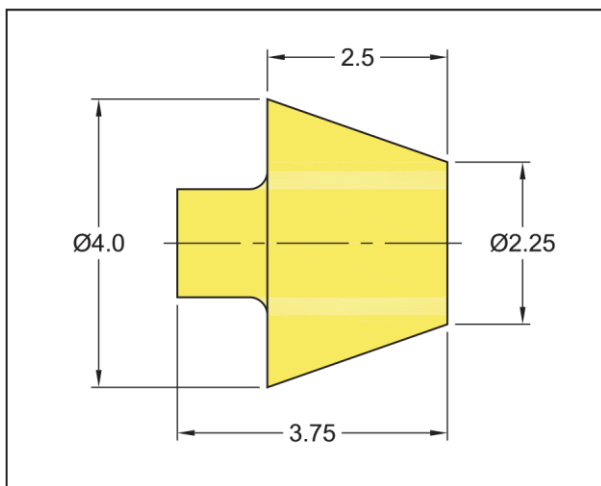


Figure 35-4

Example of G90 cycle in taper cutting - program O3503

In order to distinguish between straight cutting and taper cutting methods, using the same G90 cycle, there must be a way to distinguish these two kinds of cut, and there is one as part of the cycle parameters.

Main difference is the addition of an I(R) parameter to the cycle call, indicating the *taper amount* and its *direction per side*. This value is called a *signed radius value*. It is an I value because of its association with the X-axis. For straight cutting, the I value will always be zero and does not have to be written in the program. Its only significance is for taper cutting, in which case it has a non-zero value - Figure 35-5.

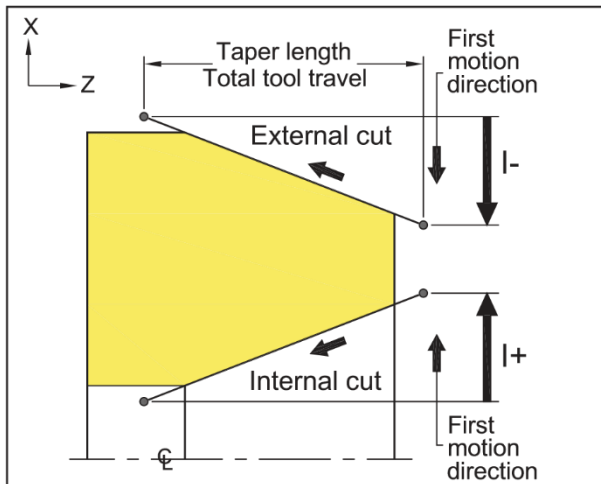


Figure 35-5

The I amount used for G90 turning cycle - external and internal

The illustration shows that the I amount is calculated as a single distance, *i.e.*, as per single side (a radius value), with specified direction, based on the *total* traveled distance and the direction of the first motion from the start position.

There are two simple rules for G90 taper cutting:

If the direction of the first tool motion in X is negative, the I amount is negative

If the direction of the first tool motion in X is positive, the I amount is positive

On a CNC lathe with the X-axis positive direction *above* the spindle center line, the typical I value will be *negative for external* taper cutting (turning) and *positive for internal* taper cutting (boring).

To program the part in Figure 35-4, keep in mind that the illustration represents the finished item and does not contain any clearances. Always add all necessary clearances first, then calculate the I amount.

In the example, a clearance of 0.100 will be added at each end of the taper, increasing its length along the axis from 2.5 to 2.7. The I amount calculation requires the *actual length* of tool travel, while maintaining the taper angle at the same time. Either the method of similar triangles or the trigonometric method can be used for such calculation (*see pages 504 and 507 for details*). Figures 35-6 and 35-7 illustrate details of the known and unknown dimensions for the I amount calculation.

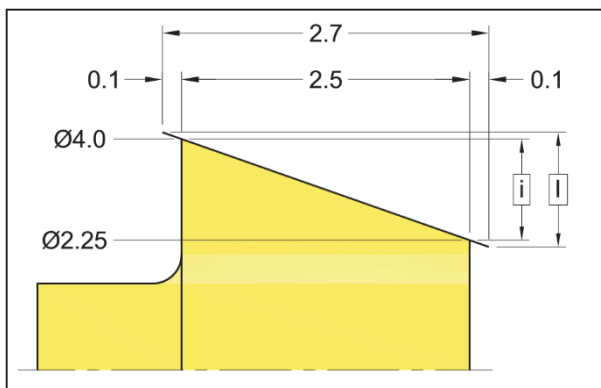


Figure 35-6

Known and unknown values for taper cutting - program O3503

Amount 'i' is known, amount 'T' has to be calculated

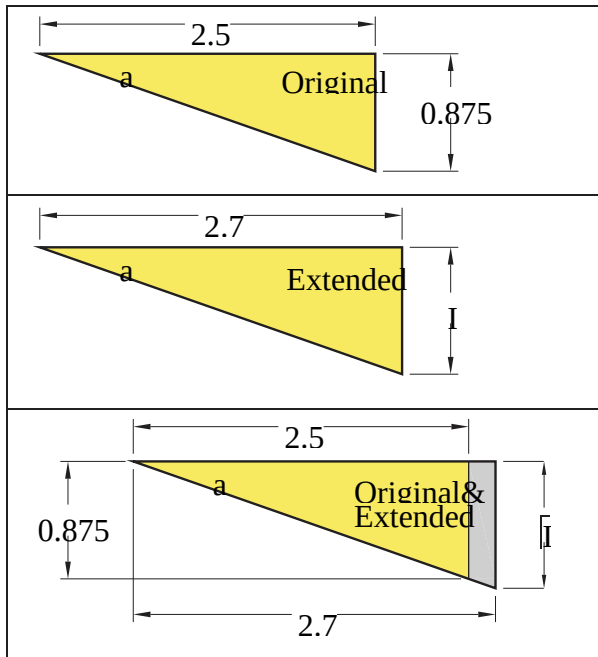


Figure 35-7

The I distance calculation using the similar triangles method

The example shown above almost suggests the simplest method of calculation, a method that is known in mathematics as the *law of similar triangles*. This law has several possible definitions, and the one that applies here is that ...

Two triangles are similar, if the corresponding sides of the two triangles are proportional

In programming, quite often there is a situation that can be solved by more than one method. Choose the one that suits better a certain programming style, then try the other method, expecting the same result. Both methods will be used here, to confirm the accuracy of the calculation.

Using Similar Triangles Method

First, calculate the difference *i* between the two known diameters, as per drawing:

$$i = (4 - 2.25) / 2 = 0.875$$

therefore, the ratio of similar triangles will be

$$I / 2.7 = i / 2.5$$

We know *i* to be 0.875, so the relations can be modified by filling in the known amount:

$$I / 2.7 = 0.875 / 2.5 \quad I = (0.875 \cdot 2.7) / 2.5$$

$$I = 0.945 \quad \dots \text{is the required amount for programming}$$

Using Trigonometric Method

The second method of calculating the I amount requires trigonometry. At this point, it is known that

$$I = 2.7 \tan a$$

and the tangent value has to be calculated first:

$$\tan a = i / 2.5 \quad \tan a = 0.875 / 2.5 \quad \tan a = 0.350$$

The amount of I can be calculated using the result:

$$I = 2.7 \cdot 0.35$$

$$I = 0.945 \quad \dots \text{is the required amount for programming}$$

In both cases, the calculations have the same result, confirming accuracy of the process. The I amount calculation is shown in *Figure 35-6* and detailed in *Figure 35-7*. Program O3503 is the final result - five cuts with 0.03 X-stock left:

O3503

(G90 TAPER TURNING EXAMPLE 1 - W/0.03 X-STOCK)

N1 G20

N2 T0100 M41

N3 G96 S450 M03

N4 G00 X4.2 Z0.1 T0101 M08 (START) N5 G90 X3.752 Z-2.6 I-0.945 F0.01 (1)

N6 X3.374 (2)

N7 X2.996 (3)

N8 X2.618 (4)

N9 X2.24 (5)

N10 G00 X10.0 Z2.0 T0100 M09 (CLEAR POS.)

N11 M01 (END OF ROUGHING)

Straight and Taper Cutting Example

Another variation of a taper is also common in CNC programming. The *Figure 35-8* shows another simple drawing, this time with a taper *and* a shoulder.

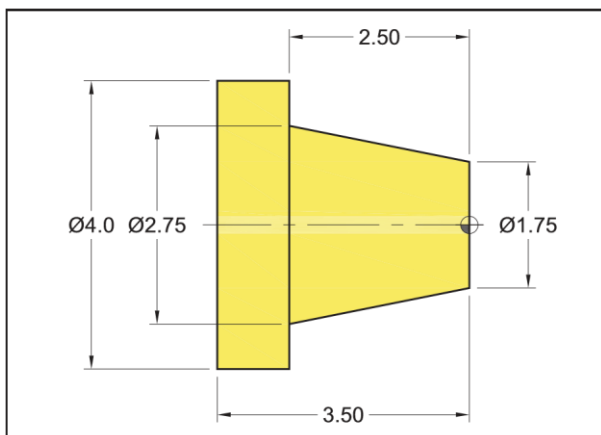


Figure 35-8

Example of G90 cycle used on a taper to a shoulder - O3504

Using the simple cycle G90, the machining requires a tapered cut towards a straight shoulder. A single G90 cycle can be used in this case as well but could result in some *excessive* or *insufficient* cutting (too much or too little stock). The best approach is to use *two modes* of the cycle - one for the straight roughing, the other for tapered roughing.

Similar to the previous example, the I taper amount has to be calculated, using the same law of similar triangles as before. The height *i* of the original triangle over the length of 2.5 is calculated as one half of the difference between the 2.750 and the 1.750:

$$i = 2.75 - 1.75 / 2 \quad i = 0.500$$

For the extended taper length, 0.005 stock amount is left at the shoulder for finishing and the taper is extended by 0.100 at the front face, for the total taper length of 2.595:

$$2.5 - 0.005 + 0.100 = 2.595$$

The I amount can now be calculated, based on the original and the extended values:

$$I / 2.595 = 0.500 / 2.5 \quad I = (0.500 - 2.595) / 2.5 \quad I = 0.519 \quad \dots \text{negative direction}$$

For roughing, a 0.030 stock will be left per side along the X axis, which is 0.060 on diameter.

In roughing operations, it is always important to select a suitable depth of cut, with safety in mind, as well as the cutting conditions. In this example, the depth of cut selection will benefit from one simple programming technique. If the depth of cut is selected arbitrarily, the *last* depth will be whatever is left to cut. A better way is to select a calculated *number of equal* cuts - *Figure 35-9*.

0.865 0.865 0.865
START
X4.100 0.161
X3.778 0.161
X3.456 0.161
X3.134 0.161
X2.812 0.173
X2.466
0.173
X2.120 0.173
X1.774

Figure 35-9

Depth of cut calculation for program example O3504

For the calculation, all that is required is to divide the distance per each side by the number of required cuts. The result will be an equal depth of cut for the whole roughing operation. If the cutting depth is too small or too large, just recalculate it with a different number of cuts. Knowing what a suitable depth of cut is a machining knowledge, expected from CNC programmers.

In *Figure 35-9*, there are 4 cuts of 0.161 for the straight roughing and 3 cuts of 0.173 for the tapered cutting. All stock allowances are in effect.

The program O3504 will use the calculations:

O3504

(G90 TAPER TURNING EXAMPLE - 2)

N1 G20

N2 T0100 M41

N3 G96 S450 M03

N4 G00 X4.1 Z0.1 T0101 M08

(START)

N5 G90 X3.778 Z-2.495 F0.01

(STRAIGHT 1)

N6 X3.456

(STRAIGHT 2)

N7 X3.134

(STRAIGHT 3)

N8 X2.812

(STRAIGHT 4)

N9 G00 X3.0 (CHANGE STRAIGHT TO TAPERED)

N10 G90 X2.812 Z-0.765 I-0.173 (TAPERED 1)

N11 Z-1.63 I-0.346 (TAPERED 2)

N12 Z-2.495 I-0.519 (TAPERED 3 - FINAL)

N13 G00 X10.0 Z2.0 T0100 M09 (CLEAR POS.)

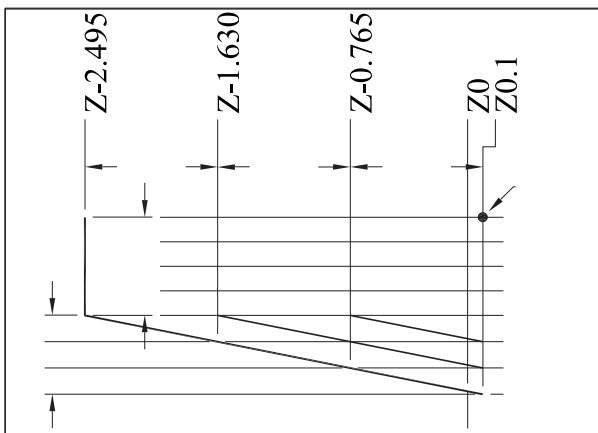
N14 M01 (END OF ROUGHING)

In a review, to calculate the amount of I or R parameter used in G90 for the taper cutting - *external* or *internal*, use the following formula:

$\frac{\text{SMALLER DIA} - \text{LARGER DIA}}{2}$

The result will also include the *sign* of the I amount.

D. G94 - FACE CUTTING CYCLE



A cycle that is very similar to G90 is another simple turning cycle, programmed with the G94 command. This cycle is called the *face cutting cycle*. The purpose of G94 cycle is to remove excessive stock between the cutter start position and the coordinates specified by the X and Z axes. The resulting cut is a straight turning cut, *normally perpendicular* to the spindle center line. In this cycle, it is the X-axis that is the main cutting direction. G94 cycle is used primarily for facing cuts and can be used for simple vertical taper cutting as well, similar to the G90 cycle.

The G94 cycle is logically identical to the G90 cycle, except the emphasis is on the X-axis cutting, rather than the Z-axis cutting

As the cycle description suggests, the G94 is normally used to perform a rough face-off of the part, towards the spindle center line or to face-off a shoulder.

Cycle Format

Similar to all cycle, the face cutting cycle G94 also has a predetermined programming format. For straight facing, the cycle format is:

G94 X (U).. Z (W).. F..

For tapered turning, the cycle format is:

G94 X (U).. Z (W).. K.. F..

Addresses X and Z are used for absolute programming, addresses U and W are used for incremental programming, and the F address is cutting feed rate. The K parameter, if greater than zero, is used for taper cutting along the vertical direction. *Figure 35-10* shows all programming parameters and cutting steps. Apply the same process as for G90 cycle.

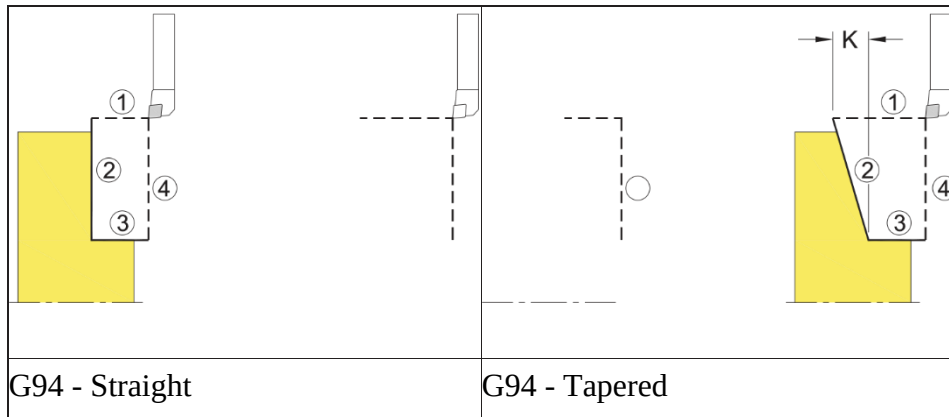


Figure 35-10

G94 turning cycle structure - straight and tapered application

E. MULTIPLE REPETITIVE CYCLES

Unlike the fixed cycles for various drilling operations on machining centers, or the G90 and G94 simple cycles for turning, the advanced cycles for CNC lathe work are much more sophisticated. The major and most distinctive feature of these cycles is their departure from the sequential order of operations. Lathe work can be very complex and the modern control systems do reflect that need. Not only straight or tapered cuts can be programmed, but also radii, chamfers, grooves, undercuts, etc.; simply, several of these cycles are used for contouring. Tool nose radius offset may also be applied, if applicable to the job.

Multiple repetitive cycles, as these cycles are called, require a computer memory in order to be useful, so the old NC machines controlled by a punched tape, *cannot* benefit from them. In tape operation, the control unit reads the tape codes sequentially, in a forward direction only. A CNC control, on the other hand, is far more complex. It can read, evaluate and process information stored in the memory in both directions, forwards and backwards - at all times. It can process mathematical instructions internally in a split of a second, simplifying the programming effort.

General Description

In total, there are seven multiple repetitive cycles available, identified by a preparatory command address G:

Profile cutting cycles - Roughing:

G71	Rough cutting cycle - <i>Horizontal emphasis</i>
G72	Rough cutting cycle - <i>Vertical emphasis</i>
G73	Pattern repeating roughing cycle

Profile cutting cycles - Finishing:

G70	Finishing cycle for G71, G72 and G73

Chip breaking cycles:

G74	Peck drilling cycle	in Z-axis - horizontal
G75	Peck grooving cycle	in X-axis - vertical

Threading cycle:

G76	Threading cycle - straight or tapered

G76 threading cycle is described separately and in sufficient detail in the threading section (see page 363).

Cycle Format Types

Each cycle is governed by very specific rules and has its *do* 's and *don* 'ts. The following sections describe each of them in detail, except the G76 threading cycle, which will be covered separately (page 363).

An important fact to take a note of, is that the format of programming for these cycles, the method of data input, is *different* for the lower level of Fanuc controls, such as the very popular 0T or the 16/18/20/21T series, than for the higher level, such as the 10/11T or the 15T series. These cycles, if they are available for the lower-level controls, require their programming format in *two blocks*, not the normal *one block*. Check the parameter settings for each control, to find about compatibility issues. Description of both formats is also included in this chapter.

Cutting Cycles and Part Contour

Probably the most common multiple repetitive cycles in turning and boring are those that are used for *profile cutting* or *contour cutting*. There are three cycles available within the roughing category: **G71, G72 and G73**

and one cycle is available for finishing:

G70

The finishing cycle is designed to finish profile generated by *any* one of the three roughing cycles.

In some respects, there is an interesting situation in programming multiple repetitive cycles. So far, the emphasis was to program roughing cuts *before* finishing cuts. This approach makes perfect sense - it is also the only logical way from the *technological* point of view. Don't be surprised if this 'rule' is suddenly broken when computer calculations take over. Yes, the implication here is that when programming the three multiple repetitive roughing cycles, the *finished contour must always be defined first*, then its machining specifications can be applied to the roughing cycle. Sounds strange? At first, perhaps. When working with these cycles longer, it will be easy to see that it is actually quite a clever and ingenious method, although hardly a recent breakthrough.

Chip breaking Cycles

The two remaining 'chip breaking' cycles are designed to produce an *interrupted cut*, either along the Z-axis (G75), or along the X-axis (G74). In practice, G74 cycle offers more practical applications than G75. G74 cycle allows to peck-drill on a CNC lathe. Although the need for peck drilling is much lower on a lathe than on a machining center, it is not that rare. However, G75 cycle, which is really a 'peck-grooving' cycle, is used rarely, as *it does not produce a precision groove*.

F. CONTOUR CUTTING CYCLES

By far, the contour cutting cycles (profiling cycles), are the most common cycles in CNC lathe programming. They are used for external (turning) and internal (boring) material removal, along almost any machinable contour.

Boundary Definition

The roughing cycles are based on the definition of *two* boundaries, typically called the *material* boundary, which is the outline of the blank, and the *part* boundary, which is the outline of the part contour. This is not a new concept at all, several early programming languages were using this method, such as the Compact II, a very popular language-based programming system of the 1970's.

The two defined boundaries create a fully enclosed area that defines the excessive material.

From this isolated area, the material is removed in an orderly way, following specified machining parameters in the cycle call block or blocks. Mathematically, the minimum number of points that can define an area is *three*. These three points must be nonlinear (meaning not on the same line). *Figure 35-11* shows a simple boundary with only three points and a boundary consisting of many points.

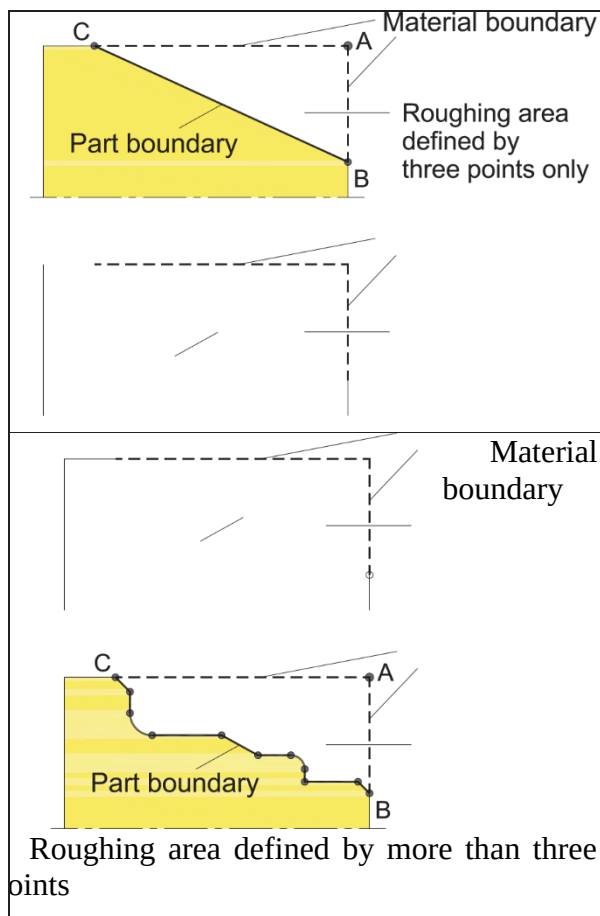


Figure 35-11

Material and part boundaries as applied to turning

In the contour cutting cycles, each point represents a tool position and the points A, B, and C represent the extreme corners of the selected (defined) machining area.

Material boundary is not defined directly, it is only implied. It lies between points A and B, and points A and C. *Material boundary can not contain* any other points; it must be a straight line, but not always a line parallel to an axis.

Part boundary is defined between points B and C and *may have* any number of points between. For CNC programming, different descriptions will be used rather than the generic ABC points in *Figure 35-11*.

Start Point and the Points P and Q

The illustrated point A is the *start point* of any contour cutting cycle. It can be defined in simple terms:

Start point is defined as the last XZ coordinate location of the tool, before the contour cutting cycle is called

Typically, this start point will be *closest* to the part corner where the rough cutting begins. It is important to select the start point very carefully, because it is more than 'just a start point'. In fact, this special point controls all approach clearances and the actual depth of the *first* roughing cut.

The generic points B and C in the last illustration will become points P and Q in the program,

respectively:

Point P represents the block number of the first XZ coordinate of the finished contour

Point Q represents the block number of the last XZ coordinate of the finished contour

Other in-depth considerations relating to P and Q boundary points are equally important - there are quite a few of them:

A number of points may be defined between the P and Q points, representing XZ coordinates of the finished contour. Contour is programmed using G01, G02, and G03 tool motions, as well as feed rates

Material removal defined by the starting point and the P-Q contour must include all necessary clearances

Tool nose radius offset should not be included between the P and Q points, but programmed *before* the cycle is called, usually during motion to the start point

For roughing, material to be machined will be divided into a series of machinable cuts. Each roughing cycle accepts a number of user supplied cutting parameters

For safety reasons, diameter of the start point should be *above* the stock diameter for external cutting, and *below* the core diameter for internal cutting

Tool motion between P point and Q point must be steadily increasing for external cutting, or steadily decreasing for internal cutting

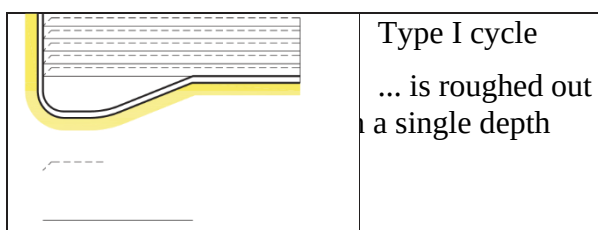
Any change in direction between P and Q points is allowed only if Type II cycle is available *and* programmed, and then in one direction only - see the next section for details

Blocks representing the first XZ coordinate of the contour P, and the last XZ coordinate of the contour Q, must contain a sequence number N, not duplicated anywhere else in the program

G. TYPE I AND TYPE II CYCLES

In the initial versions of contour cutting cycles, a change of the contouring direction into the *opposite* direction along one axis was not allowed. That limited these cycles to some extent, because common undercuts or recesses were not possible to use in the program, yet they were common in machine shops.

Presently, this older method is called *Type I* repetitive cycles. The modern controls use many more advanced software features and the change in a *single* direction is now allowed. This newer method is now called *Type II*, allowing more programming flexibility when cutting recesses and cavities (undercuts). *Figure 35-12* compares the two types and shows a disallowed contour change in two directions, within a cycle. The example applies to G71 external cutting cycle but can be modified for any internal cutting.



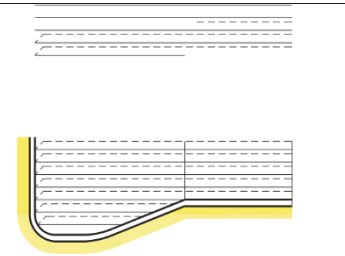
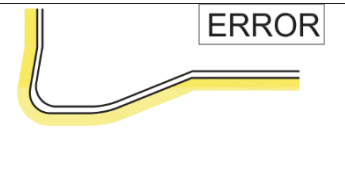
	Type II cycle ... is roughed out in several depths
	Bi-directional ... contour change in two axes NOT allowed

Figure 35-12

Comparison of Type I and Type II cycles -

- bi-directional change along two axes are not allowed

Type I allows a steadily *increasing* contour (for external cutting) or steadily *decreasing* contour (for internal cutting) from point P to point Q (typical cutting directions). On older controls, opposite X or Z direction is not allowed. Modern controls do allow an undercut to be machined with *Type I*, but the recess cutting will be done with a single pass. That may cause some heavy metal removal in certain cases. Make sure to know exactly which type the control system supports. Some experimentation may be necessary.

Type II allows a continually increasing profile or continually decreasing profile from point P to point Q. A change

into the opposite direction is allowed for a *single axis only*, depending on the active cycle. Rough out process of an undercut will be a multiple tool path. The selection of *Type I* or *Type II* is applicable to the cycle, by programming *both axes* in the block represented by the P point. This is typically the block immediately following the cycle call in the program (after G71, G72, etc.).

Programming Type I and Type II Cycles

If the control system supports *Type II* metal removal in turning and boring cycles, it also supports *Type I*, if it needs to be used for some special applications. That means, Fanuc has not replaced one type for another, it has added the *Type II*. Of course, the question is - how to distinguish between the two types in the program? Key to the type of selection is in the contents of the block that *immediately follows the cycle call*:

Type I... only one axis is specified

Type II ... two axes are specified

Example - Type I:

G71 U.. R..

G71 P10 Q.. U.. W.. F.. S..

N10 G00 X.. (ONE AXIS FOR TYPE I)

—

...

Example - Type II:

G71 U.. R..

G71 P10 Q.. U.. W.. F.. S.. N10 G00 X.. Z.. (TWO AXES FOR TYPE II)

—

...

If there is no motion along the Z-axis in the first block after the cycle call and *Type II* cycle is still required, just program W0 as the second axis.

Cycle Formatting

On the next few pages is a description of the six turning cycles, covered in detail. It is important to understand the format of each cycle as it applies to a particular control. Several Fanuc control models are available and for the purposes of programming these multiple repetitive cycles, they can be separated into two groups:

Fanuc system 0T, 16T, 18T, 20T, 21T

Fanuc system 3T, 6T, 10T, 11T, 15T

Practically, it only means a change in the way the cycle is programmed, but the subject is also important for solving some incompatibility problems. Note that the tool function T is not specified in any of the examples, although it is also allowed as a parameter in all multiple repetitive cycles. Its only need maybe for a tool offset change.

H. G71 - STOCK REMOVAL IN TURNING

The most common roughing cycle is G71. Its purpose is to remove stock by *horizontal* cutting, primarily along the Z-axis, typically from the right to the left. It is used for roughing out material out of a solid cylinder. Like all cycles, it comes in two formats - a one-block and a double block format, depending on the control system.

G71 Cycle Format - 6T/10T/11T/15T The one-block format for the G71 cycle is:

G71 P.. Q.. I.. K.. U.. W.. D.. F.. S..

+ where ...

- P = First block number of the finishing contour
- Q = Last block number of the finishing contour
- I = Distance and direction of rough semi finishing in the X-axis - per side
- K = Distance and direction of rough semi finishing in the Z-axis
- U = Stock amount for finishing on the X-axis diameter
- W = Stock left for finishing on the Z-axis
- D = Depth of roughing cut
- F = Cutting feed rate (in/rev or mm/rev) overrides feed rates between P block and Q block
- S = Spindle speed (ft/min or m/min) overrides spindle speeds between P block and Q block

The I and K parameters are not available on all machines. They control the amount of cut for semi finishing, the last continuous cut before final roughing motions.

G71 Cycle Format - 0T/16T/18T/20T/21T

If the control requires a double block entry for the G71 cycle, the programming format is:

G71 U.. R..

G71 P.. Q.. U.. W.. F.. S..

+ where ...

First block:

U = Depth of roughing cut R = Amount of retract from each cut *Second block:*

P First block number of the finishing
= itour

Q Last block number of the finishing
= itour

U Stock amount for finishing on the
= axis diameter

W Stock left for finishing on the Z-
= s

F Cutting feed rate (in/rev or
= n/rev) overrides

feed rates between P block and Q block

S = Spindle speed (ft/min or m/min) overrides spindle speeds between P block and Q block

Do not confuse address U in the first block, depth of cut per side, and address U in the second block, stock left on diameter. The I and K parameters may be used only on some controls and the retract amount R is set by a system parameter.

Example of external and internal use of the G71 cycle will use drawing data in *Figure 35-13*. The examples will use the two-block G71 method, as it is more common.

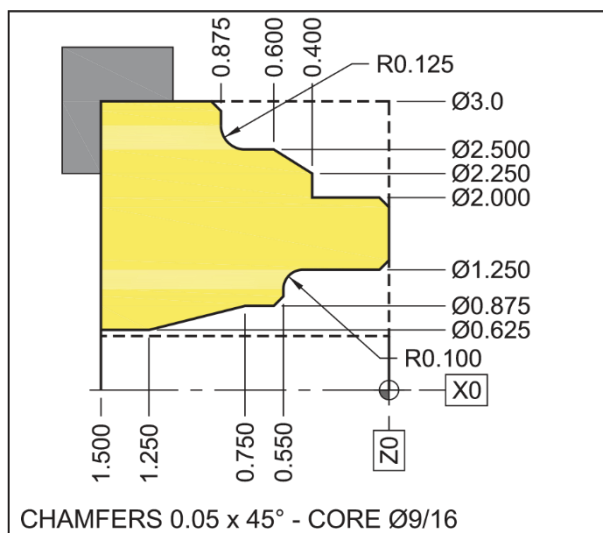


Figure 35-13

Drawing example to illustrate G71 roughing cycle - program O3505

G71 for External Roughing

The stock material in the example has an existing hole diameter 0.5625 (9/16). For external cutting of this part, a standard 80 tool will be used for a single cut on the face, as well as for roughing the outer contour. Program O3505 covers these operations.

O3505 (G71 ROUGHING CYCLE - ROUGHING ONLY)

N1 G20

N2 T0100 M41 (OD ROUGHING TOOL + GEAR)

N3 G96 S450 M03 (SPEED FOR ROUGH TURNING)

N4 G00 G41 X3.2 Z0 T0101 M08 (START FOR FACE)

N5 G01 X0.36 (END OF FACE DIA) N6 G00 Z0.1 (CLEAR OFF FACE)

N7 G42 X3.1 (START POSITION FOR CYCLE)
N8 G71 U0.125 R0.04
N9 G71 P10 Q18 U0.06 W0.004 F0.014
N10 G00 X1.7 (P POINT = START OF CONTOUR)
N11 G01 X2.0 Z-0.05 F0.005
N12 Z-0.4 F0.01
N13 X2.25
N14 X2.5 Z-0.6
N15 Z-0.875 R0.125
N16 X2.9
N17 G01 X3.05 Z-0.95
N18 U0.2 F0.02 (Q POINT = END OF CONTOUR)
N19 G00 G40 X5.0 Z6.0 T0100 N20 M01

External roughing has been completed at this point in the program and internal roughing can be programmed for the next tool. In all examples that include a tool change between a short tool (such as a turning tool) and a long tool (such as a boring bar), it is important to move the short tool further from the front face. The motion should be far enough to accommodate the incoming long tool. The clearance is 6.0 in the above example (block N18 with Z6.0).

G71 for Internal Roughing

The facing cut has been done with the previous tool and the roughing boring bar can continue machining:

N21 T0300 (ID ROUGHING TOOL)
N22 G96 S400 M03 (SPEED FOR ROUGH BORING)
N23 G00 G41 X0.5 Z0.1 T0303 M08 (START POS.)
N24 G71 U0.1 R0.04
N25 G71 P26 Q33 U-0.06 W0.004 F0.012
N26 G00 X1.55 (P POINT = START OF CONTOUR)
N27 G01 X1.25 Z-0.05 F0.004
N28 Z-0.55 R-0.1 F0.008
N29 X0.875 K-0.05
N30 Z-0.75
N31 X0.625 Z-1.25
N32 Z-1.55
N33 U-0.2 F0.02 (Q POINT = END OF CONTOUR)
N34 G00 G40 X5.0 Z2.0 T0300 N35 M01

Now, the part has been completely roughed out, leaving only the required stock on diameters and faces or shoulders. Finishing with G70 cycle, described later, is possible with the same tool, if tolerances and/or surface finish are not too critical. Otherwise, another tool or tools will be required in the same program, after a tool change.

At this stage, evaluate what has been done and why. Many principles that applied to the example

are very common to other operations that also use the multiple repetitive cycles. It is important to learn them well at this point.

Direction of Cutting in G71

Example O3505 shows that G71 can be used for roughing *externally* or *internally* with two important differences:

Start point relative to P point (SP to P versus P to SP)

Sign of U address for stock allowance on diameter

Control system will process the cycle for *external* cutting, if the X direction from the start point SP to the point P is *negative*. In the example, X start point is X3.1, P point is X1.7. The X direction is *negative* or decreasing and *external* cutting will take place.

Control system will process the cycle for *internal* cutting, if the X direction from the start point SP to the point P is *positive*. In the example, X start point is X0.5, P point is X1.55. The X direction is *positive* or increasing, and *internal* cutting will take place.

Figure 35-14 illustrates the concept of G71 cycle, as applied to both, external and internal cutting.

By the way, although the *sign* of the stock U value is very important for the final size of the part, it does *not* determine the mode of cutting. This concludes the section relating to the G71 multiple repetitive cycle. The face roughing cycle G72 is similar and is described next.

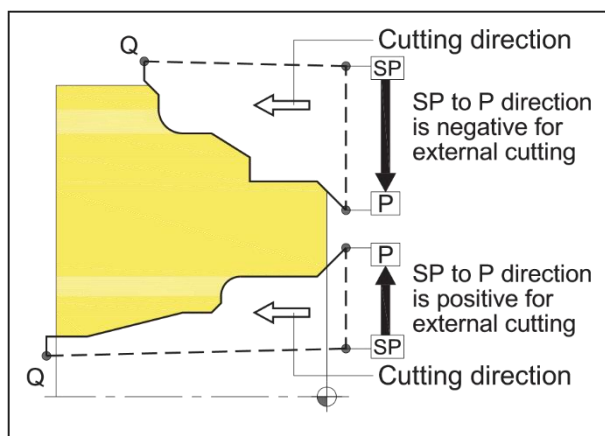


Figure 35-14

External and internal cutting in G71 cycle

I. G72 - STOCK REMOVAL IN FACING

G72 cycle is identical in every respect to the G71 cycle, except the stock is removed mainly by *vertical* cutting (facing), typically from the large diameter towards the spindle center line X0. It is used for roughing of a solid cylinder, using a series of *vertical* cuts (face cuts). Like all other cycles in this group, it comes in two formats - a one block and a double block format, depending on the control system. Compare G72 with the G71 structure on examples in this chapter and you will find many similarities.

G72 Cycle Format - 6T/10T/11T/15T

The one-block programming format for the G72 cycle is:

G72 P.. Q.. I.. K.. U.. W.. D.. F.. S..

+ where ...

= First block number of the finishing contour

= Last block number of the finishing contour

I = Distance and direction of rough semi finishing in the X-axis - per side

K = Distance and direction of rough semi finishing in the Z-axis

U = Stock amount for finishing on the X-axis diameter

W = Stock left for finishing on the Z-axis

D = Depth of roughing cut

F = Cutting feed rate (in/rev or mm/rev) overrides feed rates between P block and Q block

S = Spindle speed (ft/min or m/min) overrides spindle speeds between P block and Q block

Meaning of each address is the same as for the G71 cycle. The I and K parameters are not available on all machines. These parameters control the amount of cut for semi finishing, which is the last continuous cut before final roughing motions are completed.

G72 Cycle Format - 0T/16T/18T/20T/21T

If the control system requires a double block entry for the G72 cycle, the programming format is:

G72 W.. R..

G72 P.. Q.. U.. W.. F.. S..

+ where ...

First block:

W = Depth of roughing cut R = Amount of retract from each cut *Second block:*

P First block number of the finishing
= contour

Q Last block number of the finishing
= contour

U Stock amount for finishing on the
= axis diameter

W Stock left for finishing on the Z-
= s

F Cutting feed rate (in/rev or
= n/rev) overrides feed rates between
P block and Q block

S Spindle speed (ft/min or m/min)
= overrides spindle speeds between P
block and Q block

In G71 cycle for the double block definition, there were two addresses U. In G72 double block definition cycle, there are two addresses W. Make sure you do not confuse the W in the first block - depth of cut (actually it is a *width* of cut), and the W in the second block - stock left on faces. The I and K parameters may be available, depending on the control.

An example program O3506 for the G72 cycle uses the drawing data in *Figure 35-15*.

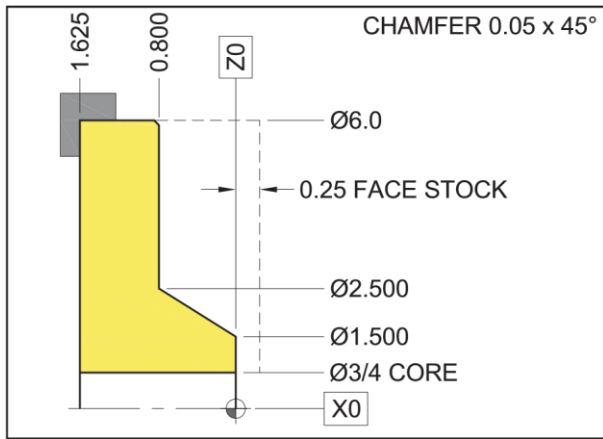


Figure 35-15

Drawing example to illustrate G72 roughing cycle - program O3506

In this facing application, all main data will be reversed by 90, because the cut will be segmented along the X-axis. Roughing program using the G72 cycle is logically similar to the G71 cycle:

O3506 (G72 ROUGHING CYCLE - ROUGHING ONLY)

N1 G20

N2 T0100 M41 (OD FACING TOOL + GEAR)

N3 G96 S450 M03 (SPEED FOR ROUGH FACING)

N4 G00 G41 X6.25 Z0.3 T0101 M08 (START POS.)

N5 G72 W0.125 R0.04

N6 G72 P7 Q13 U0.06 W0.03 F0.014

N7 G00 Z-0.875 (P-POINT = START OF CONTOUR)

N8 G01 X6.05 F0.02

N9 X5.9 Z-0.8 F0.008

N10 X2.5

N11 X1.5 Z0

N12 X0.55

N13 W0.1 F0.02 (Q-POINT = END OF CONTOUR)

N14 G00 G40 X8.0 Z3.0 T0100 N15 M01

General concept of G72 cycle is illustrated in Figure 35-16. Note the position of point P as it relates to the start point SP and compare it with the G71 cycle.

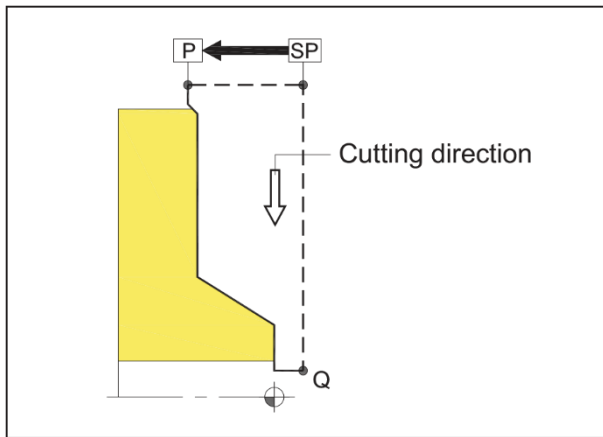


Figure 35-16

Basic concept of G72 multiple repetitive cycle

J. G73 - PATTERN REPEATING CYCLE

The pattern repeating cycle is also called the *Closed Loop* or a *Profile Copying* cycle. Its purpose is to minimize the cutting time for roughing material of irregular shapes and forms, for example, forgings and castings.

G73 Cycle Format - 6T/10T/11T/15T

The one-block programming format for G73 cycle is similar to both the G71 and G72 cycles:

G73 P.. Q.. I.. K.. U.. W.. D.. F.. S..

+ where ...

= First block number of finishing contour

= Last block number of finishing contour

I = X-axis distance and direction of relief - per side

K = Z-axis distance and direction of relief

U = Stock amount for finishing on the X-axis diameter

W Stock left for finishing on the Z-axis

D The number of cutting divisions

F Cutting feed rate (in/rev or mm/rev) overrides feed rates between P block and Q block

S Spindle speed (ft/min or m/min) overrides spindle speeds between P block and Q block

G73 Cycle Format - 0T/16T/18T/20T/21T

If your control system requires a double block entry for the G73 cycle, the programming format is:

G73 U.. W.. R..

G73 P.. Q.. U.. W.. F.. S..

+ where ...

First block:

U = X-axis distance and direction of relief - per side

W = Z-axis distance and direction of relief R = Number of cutting divisions

Second block:

P First block number of the finishing
= 1st tour

Q Last block number of the finishing
= 1st tour

U Stock amount for finishing on the
= axis diameter

W Stock left for finishing on the Z-
= s

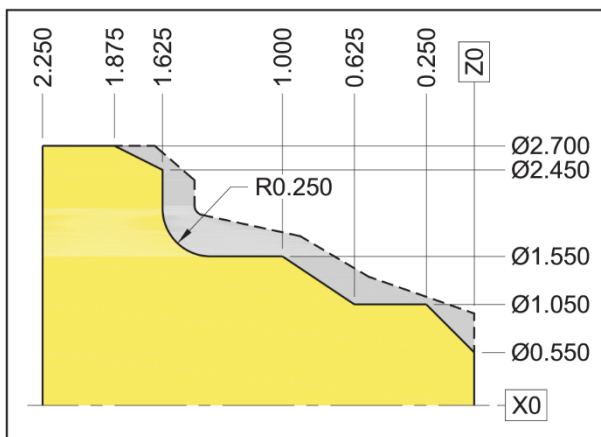
F Cutting feed rate (in/rev or
= n/rev) overrides feed rates between
block and Q block

S Spindle speed (ft/min or m/min)
= overrides spindle speeds between P
block and Q block

In the two-block cycle entries, do not mix up addresses in the first block that repeat in the second block (U and W in the G73 example). *They have a different meaning!*

G73 Example of Pattern Repeating

Pattern repeating cycle G73 program example uses the drawing in *Figure 35-17*.



There are three important input parameters in G73 cycle U/W/R (I/KD). One parameter seems to be missing - *there is no depth of cut specification!* In G73 cycle, it is not needed. The actual depth of cut is calculated automatically, based on these three parameters:

U (I) ... amount of rough material to remove in X-axis w (K) ... amount of rough material to remove in Z-axis

R (D) ... number of cutting divisions or repeats

Use this cycle with care - its design assumes an *equal amount* of rough stock to be removed along both the X and the Z axes. That is not the typical reality for forgings and castings, where the stock *varies* all over the material - see the illustration in *Figure 37-17*. The cycle can still be used with a reasonable efficiency, but some 'air' cutting may be an unwanted side effect for odd, shaped parts.

In the example, the *largest expected* material amount per one side will be chosen as 0.200 (U0.2) and the *largest expected* material amount on the face as 0.300 (W0.3). The number of divisions could be either two or three, so the program will use R3. Some modification at the control may be necessary during actual setup or machining, depending on the exact condition and sizes of the casting or forging.

This cycle is suitable for roughing contours where the finish contour closely matches the contour of the casting or forging. Even if there is some 'air' cutting, this cycle may be more efficient than the selection of the G71 or G72 cycles where too much empty cutting would take place. The program O3507 shows roughing and finishing with the same tool (as an example):

O3507 (G73 PATTERN REPEATING CYCLE)

N1 G20 M42

N2 T0100

N3 G96 S350 M03

N4 G00 G42 X3.0 Z0.1 T0101 M08 N5 G73 U0.2 W0.3 R3

Figure 35-17

Pattern repeating cycle G73 - program example O3507

N6 G73 P7 Q14 U0.06 W0.004 F0.01

N7 G00 X0.35

N8 G01 X1.05 Z-0.25

N9 Z-0.625

N10 X1.55 Z-1.0

N11 Z-1.625 R0.25

N12 X2.45

N13 X2.75 Z-1.95

N14 U0.2 F0.02

N15 G70 P6 Q13 F0.006

N16 G00 G40 X5.0 Z2.0 T0100

N17 M30

%

The number of passes R3 may be necessary to accommodate some rotating eccentricity, normally associated with castings and forgings. On the other hand, R2 may be necessary for the heavier cut, to 'bite under the skin' of the material, for better tool life. Schematically, *Figure 35-18* shows three programmed cutting divisions (R3).

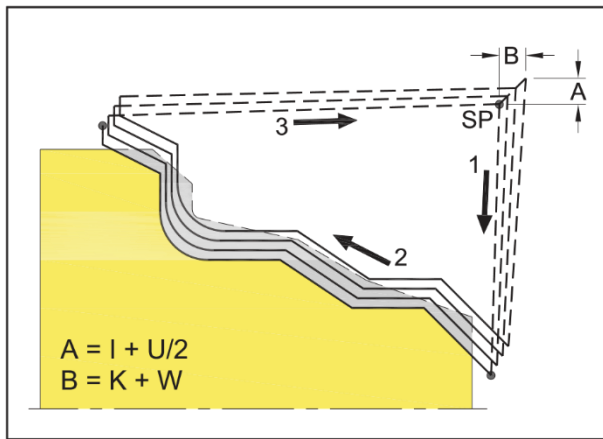


Figure 35-18

Schematic representation of G73 cycle

Note that the pattern repeating cycle does exactly that - it repeats the machining contour (pattern) specified between the P and Q points. Each individual tool path is offset by a calculated amount along the X and Z axes. On the machine, watch cutting progress with care - particularly for the first tool path. Feed rate override may come useful here.

K. G70 - CONTOUR FINISHING CYCLE

The last of the contouring cycles is G70. Although it has a smaller G number than any of the three roughing cycles G71, G72 and G73, finishing cycle G70 is normally used *after* any one of these three roughing cycles. As its description suggests, it is strictly used *for finishing cut of a previously defined contour*.

G70 Cycle Format - All Controls

For this cycle, there is no difference in the programming format for various controls - it is all the same, and the cycle call is a one-block command.

The programming format for G70 cycle is:

G70 P.. Q.. F.. S..

+ where ...

P = First block number of the
ishing contour

Q = Last block number of the
ishing contour

F = Cutting feed rate (in/rev or
n/rev)

S = Spindle speed (ft/min or
min)

Cycle G70 accepts a previously defined finishing contour from either of the three roughing cycles, already described. This finishing contour is defined by the P and Q points of the respective cycles, and is normally repeated in the G70 cycle, although other block number references may be used - **BE CAREFUL HERE!**

For safety, always use the same start point for G70 as for the roughing cycles

Earlier roughing program O3505, using the G71 repetitive cycle for rough turning and rough boring, can be completed by using another two tools, one for external, one for internal finishing

tool path:

(O3505 CONTINUED ...) ...

N36 T0500 M42 (OD FINISHING TOOL + GEAR)

N37 G96 S530 M03 (SPEED FOR FINISH TURNING)

N38 G42 X3.1 Z0.1 T0505 M08 (START POS.)

N39 G70 P10 Q18 (FINISHING CYCLE - OD)

N40 G00 G40 X5.0 Z6.0 T0500 N41 M01

N42 T0700 (ID FINISHING TOOL)

N43 G96 S475 M03 (SPEED FOR ROUGH BORING)

N44 G00 G41 X0.5 Z0.1 T0707 M08 (START POS.)

N45 G70 P26 Q33 (FINISHING CYCLE - ID)

N46 G00 G40 X5.0 Z2.0 T0700

N47 M30 (END OF PROGRAM)

%

Even for external finishing, the cutting tool is still programmed to start *above* the original stock diameter and off the front face, although all roughing motions have already been completed. A similar approach applies to the internal cut. For safety reasons, this is the recommend practice.

There are no feed rates programmed for G70 cycle, although the cycle format does accept a feed rate. The defined block segments P to Q for roughing tool already include feed rates. These programmed feed rates will be ignored in roughing mode and will become active only for G70 cycle, during finishing. If the finish contour did not include any feed rates, then program a *common feed rate* for finishing all contours during the G70 cycle processing. For example, program block

N39 G70 P10 Q18 F0.007

will be a waste of time, since the 0.007 in/rev feed rate will *never* be used. It will be *overridden* by the feed rate defined between blocks N9 and N17 of program O3505. On the other hand, if there is no feed rate programmed for the finishing contour at all, then

N.. G70 P.. Q.. F0.007

will use 0.007 in/rev exclusively for the finishing tool path.

The same logic described for G71 cycle, applies equally to G72 cycle.

Roughing program O3506, using G72 cycle for rough turning of the part face, can be completed by using another external tool for finishing cuts using G70 cycle:

(O3506 CONTINUED ...) ...

N16 T0500 M42 (OD FACING TOOL + GEAR)

N17 G96 S500 M03 (SPEED FOR FINISH FACING)

N18 G00 G41 X6.25 Z0.3 T0505 M08 (START POS.)

N19 G70 P7 Q13 (FINISHING CYCLE)

N20 G00 G40 X8.0 Z3.0 T0500

N21 M30

%

The rules mentioned earlier also apply for contour finishing defined by G72 cycle. Program

O3507, using the G73 cycle, can also be programmed by using another external tool for finishing, applying the same rules.

L. BASIC RULES FOR G70-G73 CYCLES

In order to make the multiple repetitive stock removal cycles (contouring cycles) work properly and efficiently, observing the rules of their use is very important. Often a small oversight may cause a lengthy delay.

Here are the most important rules and observations as they relate to G71, G72 and G73 turning/boring cycles:

Always apply tool nose radius offset before the stock removal cycle is called

Always cancel tool nose radius offset after the stock removal cycle is completed

Return motion to the start point is automatic, and must not be programmed

The P block in G71 should not include the Z-axis value (Z or W) for cycle Type I

Change of direction is allowed only for Type II

G71 cycle, and along one axis only (W0)

Stock allowance U is programmed on a diameter, and its sign shows to which side of the stock it is to be applied - sign is the X-direction relative to centerline: ... U+ for turning - towards spindle centerline

... U- for boring - away from spindle centerline

Feed rate programmed for the finishing contour (specified between P and Q points) will be ignored during roughing

D address in one-block format only, does not use decimal point, and must be programmed for leading zero suppression format:

D0750 or D750 is equivalent to 0.0750 depth

Only some control systems do allow a decimal point to be used for the D address (depth of cut) in one-block G71 and G72 cycles

M. G74 - PECK DRILLING CYCLE

G74 cycle is one of two cycles usually used for non-contour work. Along with G75 cycle, it is used for machining an interrupted cut, such as chips breaking during a long cutting motion. *G74 cycle is used along the Z-axis.*

This is the cycle commonly used for an interrupted cut along the Z-axis. The name of the cycle is *Peck Drilling Cycle*, similar to G73 peck drilling cycle used for machining centers. For lathe work, G74 cycle application is a little more versatile than for its G73 equivalent on machining centers. Although its main purpose may be applied towards peck drilling, the cycle can be used with equal efficiency for interrupted cuts in non-contour turning and boring (for example, in some very hard materials), deep face grooving, difficult part-off machining, and many other applications.

G74 Cycle Format - 6T/10T/11T/15T

The one-block programming format for G74 cycle is:

G74 X.. (U..) Z.. (W..) I.. K.. D.. F.. S..

+ where ...

X(U) = Final groove diameter to be cut

Z(W) = Z-position of the last peck - depth of hole
 I = Depth of each cut (no sign)
 K = Distance of each peck (no sign)
 D = Relief amount at the end of cut

(Must be zero for face grooving)

F = Groove cutting feed rate (in/rev or mm/rev)
 S = Spindle speed (ft/min or m/min)

G74 Cycle Format - 0T/16T/18T/20T/21T

The two-block programming format for G74 cycle is:

G74 R..

G74 X.. (U..) Z.. (W..) P.. Q.. R.. F.. S..

+ where ...

First block:

R = Return amount (relief clearance for each cut)

Second block:

X(U) = Final groove diameter to be cut

Z(W) = Z-position of the last peck - depth of hole

= Depth of each cut (no sign)

= Distance of each peck (no sign)

= Relief amount at the end of cut

(Must be zero for face grooving)

F = Groove cutting feed rate (in/rev or mm/rev)

S = Spindle speed (ft/min or m/min)

If both the X/U and P (or I) are omitted in the cycle, the machining is along the Z-axis only (peck drilling). In a typical peck drilling operation, only the Z, K and F values are programmed - see Figure 35-19.

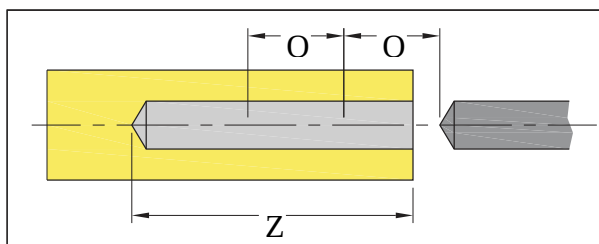


Figure 35-19

Schematic format for G74 cycle example (two block format)

The following program example illustrates G74 cycle:

O3507 (G74 PECK DRILLING)

N1 G20

N2 T0200

N3 G97 S1200 M03	(SPEED IN RPM)
N4 G00 X0 Z0.2 T0202 M08 N5 G74 R0.02	(START POSITION)
N6 G74 Z-3.0 Q6250 F0.012	(PECK DRILLING)
N7 G00 X6.0 Z2.0 T0200	(CLEAR POSITION)
N8 M30 %	(END OF PROGRAM)

Drilling will take place to a three-inch depth, in depth increments of 0.625 of an inch. Note the first depth peck is calculated from the start position. Programming of an interrupted groove is very similar in format.

N. G75 - GROOVE CUTTING CYCLE

The G75 cycle is the other of two lathe cycles available for simple, non-precision work. Together with the G74 cycle, it is used for operations requiring an interrupted cut, for example for breaking chips during a long or deep cutting motion. *G75 cycle is used along the X axis.*

This is also a very simple cycle, designed to break chips during a rough cut along the X axis - used mainly for a grooving operation. The G75 cycle is identical to G74, except the X axis is replaced with the Z axis.

G75 Cycle Format - 6T/10T/11T/15T

The one-block programming format for G75 cycle is:

G75 X.. (U..) Z.. (W..) I.. K.. D.. F.. S..

+ where ...

X(U) = Final groove diameter to be cut

Z(W) = Z-position of the last groove

(For multiple grooves only)

I = Depth of each cut (no sign)

K = Distance between grooves (no sign)

(For multiple grooves only)

D = Relief amount at the end of cut

(Must be zero or not used for face groove)

F = Groove cutting feed rate (in/rev or mm/rev)

S = Spindle speed (ft/min or m/min)

G75 Cycle Format - 0T/16T/18T/20T/21T

The two-block programming format for G75 cycle is:

G75 R..

G75 X.. (U..) Z.. (W..) P.. Q.. R.. F.. S..

+ where ...

First block:

R = Return amount (relief clearance for each cut)

Second block:

X(U) = Final groove diameter to be cut

Z(W) = Z-position of the last groove
= Depth of each cut (no sign)
= Distance between grooves (no sign)
= Relief amount at the end of cut

(Must be zero for face grooving)

F = Groove cutting feed rate
(Usually in/rev or mm/rev)

S = Spindle speed (usually ft/min or m/min)

If both the Z/W and Q (or K) are omitted in the cycle, the machining is along the X-axis only (peck grooving).

A practical example of G75 cycle will be presented in the next chapter.

O. BASIC RULES FOR G74 AND G75 CYCLES

Several notes are common to both cycles:

In both cycles, the X and Z values can be programmed either in the absolute or incremental mode.

Both cycles allow an automatic relief.

Relief amount at the end of cut can be omitted in that case it will be assumed as zero.

Return amount (clearance for each cut) is only programmable for the two-block method. Otherwise, it is set by an internal parameter of the control system.

If the return amount is programmed (two-block method), and the relief amount is also programmed, the presence of X determines the meaning. If the X-value is programmed, the R-value means the relief amount.

28. GROOVING ON LATHES

Groove cutting on CNC lathes is a multi-step machining operation. The term *grooving* usually applies to a process of forming a narrow cavity of a certain depth, on a cylinder, cone, or a face of the part. The groove shape, or at least a significant part of it, will be in the shape of the cutting tool. Grooving tools are also used for a variety of special machining operations.

Grooving tool is usually a carbide insert mounted in a special tool holder, similar to any other tool. Designs of grooving inserts vary, from a single tip to an insert with multiple tips. Inserts are manufactured to nominal sizes. Multi tip insert grooving tools are used to decrease costs and increase productivity.

A. GROOVING OPERATIONS

Cutting tools for grooving are either *external* or *internal* and use a variety of inserts in different configurations. The most important difference between grooving and turning is the *direction of cut*. Turning tool can be applied for cuts in multiple directions, grooving tool is normally used to cut in a single direction only. A notable exception is an operation known as necking (relief grooving), which takes place at 45°, where the cutting insert angle and the infeed angle *must be identical* (usually at 45°). There is another application of a two-axis simultaneous motion in grooving, a corner breaking on the groove. Strictly speaking, this is a turning operation. Although a grooving tool is not designed for turning, it can be used for some light machining, like cutting a small chamfer. During the corner breaking cut on a groove, the amount of material removal is always very small, and the applied feed rate is normally low.

Main Grooving Applications

Groove is an essential part of components machined on CNC lathes. There are many kinds of grooves used in industry. Most likely, programming will include many undercuts, clearance and recess grooves, oil grooves, etc. Some of the main purposes of grooving are to allow two components to fit face-to-face (or shoulder-to-shoulder) and, in case of lubrication grooves, to let oil or some other lubricant to flow smoothly between two or more connecting parts. There are also pulley or V-belt grooves that are used for belts to drive a motor. O-ring grooves are specially designed for insertion of metal or rubber rings, that serve as stoppers or sealers. Many industries use grooves unique to their needs, others use the more general groove types.

Grooving Criteria

For a CNC programmer, grooving usually presents no special difficulties. Some grooves may be easier to program than others, yet there could be several fairly complex grooves found in various industries that may present a programming or machining challenge. In any case, before a groove can be programmed, have a good look at the drawing specifications and do some overall evaluations. Many grooves may appear on the same part at different locations and could benefit from a subprogram development. When planning a program for grooving, evaluate each groove carefully. In good program planning, evaluate the selected groove by at least three criteria:

Groove shape

Groove location on a part

Groove dimensions and tolerances

Unfortunately, many grooves are not of the highest quality. Perhaps it is because many grooves do not require high precision and when a high precision groove has to be done, the programmer does not know how to handle it properly. Watch particularly for surface finish and tolerances.

B. GROOVE SHAPE

The first evaluation before programming grooves is the groove shape. Shape is determined by the part drawing and corresponds to the groove purpose. Groove shape is the single most important

factor when selecting a grooving insert. A groove with sharp corners parallel to the machine axes requires a square insert, a groove with radius requires an insert having the same or smaller radius. Special purpose grooves, for example an angular groove shape, will need an insert with angles corresponding to the groove angles as given in the drawing. Formed grooves require inserts shaped into the same form, etc. Some typical shapes of grooving inserts are illustrated in *Figure 36-1*.

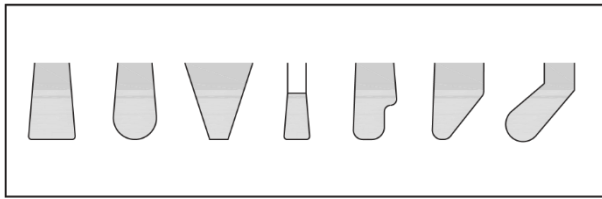


Figure 36-1

Typical shapes of common grooving tools

333

Nominal Insert Size

In many groove cutting operations, the groove width will be greater than the largest available grooving inserts of a nominal size (*i.e.*, off the shelf size). Nominal sizes are normally found in various tooling catalogues and typically have widths like 1 mm, 2 mm, 3 mm or 1/32, 3/64, 1/16, 1/8 in inches, and so on, depending on the units selected.

For example, a groove width of 0.276 inches can be cut with a nearest *lower* nominal insert width of 0.250 inch. In such cases, the groove program has to include at least two cuts - one or more roughing cuts, in addition to at least one finishing cut. Another grooving tool may be used for finishing, if the tolerances or excessive tool wear make it more practical - *Figure 36-2*.

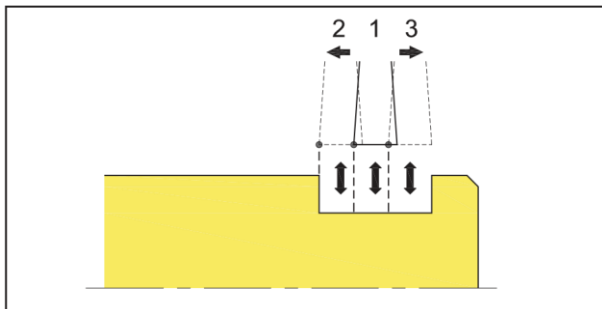


Figure 36-2

Cut distribution for grooves wider than the insert

Insert Modification

Once in a while, programmers encounter a groove that requires a special insert in terms of its size or shape. There are two options to consider. One is to have a custom-made insert, if it is possible and practical. For a large number of grooves, it may be a justified solution. The other alternative is to modify an existing insert in-house.

Generally, in CNC programming, off-the-shelf tools and inserts should be used as much as possible. In special cases, however, a standard tool or insert can be modified to suit a particular job. For grooving, it may be a small extension of the insert cutting depth, or a radius modification. Try to modify the groove shape itself only as the last resort. Modification of standard tools slows down the production and can be quite costly.

C. GROOVE LOCATION

Groove location on a part is determined by the part drawing. The locations can be one of three groups:

Groove cut on a cylinder ... diameter cutting Groove **cut on a cone ... taper cutting**

Groove cut on a face ... shoulder cutting

Although some variations are possible, for practical purposes, only these three categories are considered. Each of the three locations may be either *external* or *internal*.

The two most common groove locations are on a cylinder, *i.e.*, on a straight outside - or *external* - diameter, or on a straight inside - or *internal* - diameter. Many other grooves may be located on a face, on a taper (cone), even in a corner. The illustration in *Figure 36-3* shows some typical locations of various grooves.

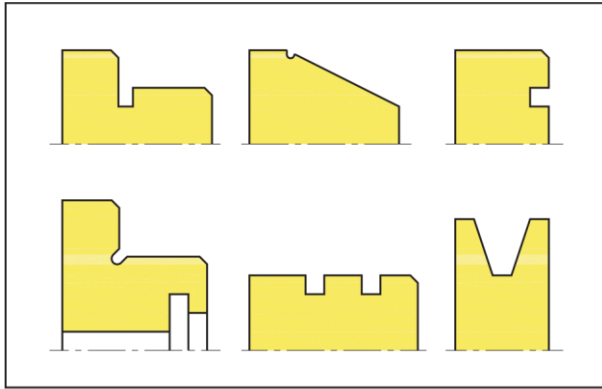


Figure 36-3

Typical groove locations on a part

D. GROOVE DIMENSIONS

Dimensions of a groove are always important when selecting the proper grooving insert. Grooving dimensions include the *width* and *depth* of a groove, as well as corner specifications. It is not possible to cut a groove with an insert that is larger than the groove width. Also, it is not possible to feed into a groove depth that is greater than the depth clearance of the insert or tool holder. However, there is usually no problem in using a narrow grooving insert to make a wide groove with multiple cuts. The same applies for a deep cutting insert used to make a shallow groove. *Dimensions of a groove determine the method of machining.* A groove whose width equals the insert width selected for the groove shape, requires only *one* cut. Simple *feed-in* and *rapid-out* tool motion is all that is required. To program a groove correctly, the *width* and *depth* of the groove must be known as well as its *position* relative to a known reference position on the part. This position is the distance to one side - or one wall - of the groove.

Some extra-large grooves require a special approach. For example, a groove that is 10 mm wide and 8 mm deep cannot be cut in a single pass. In this case, rough cuts for the groove will control not only its width, but also its depth. It is not unusual to even use more than one tool for such an operation. Grooving program may also need to be designed in sections. In case of an insert breakage, only the affected program section has to be repeated.

Groove Position

In *Figure 36-4* are shown two most common methods of dimensioning a typical groove. Groove width is given in both cases as dimension W, but the distance L from the front face is different in the example *a* and example *b*.

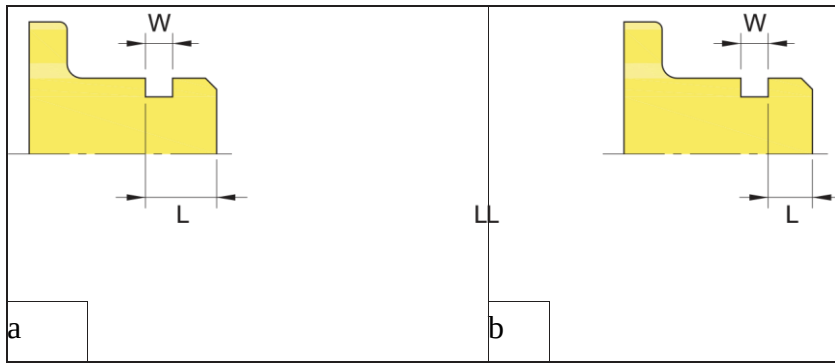


Figure 36-4

Groove position dimensioning - two common methods

In example 36-4a, the dimension L is measured to the *left* side of the groove. For programming purposes, this dimension is more convenient, because it will actually appear in the program as specified in the drawing. Normally, the standard tool reference point of a grooving tool is set to the *left side* of the grooving insert.

Figure 36-4b example shows the dimension L to the *right* side of the groove. The left side dimension can be found easily, by adding the groove width W. Programming considerations will be slightly different, particularly if the dimensional tolerances are specified. Always take a safer approach and presume that the specified dimension indicates *more important* dimension. If a tolerance range is specified for any dimension, this tolerance must always be maintained on the finished groove, and it will affect the overall programming method. A groove may also be dimensioned from another reference location, depending on its purpose.

Groove Depth

In Figure 36-5, there are two typical methods of dimensioning the groove *depth*.

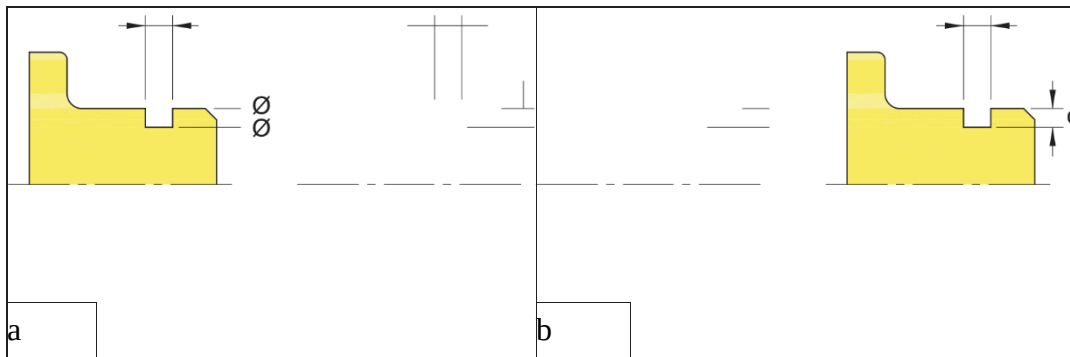


Figure 36-5

Groove depth dimensioning - two common methods

In the Figure 36-5a, the top and bottom groove diameter are both given. This method has a major benefit that the bottom groove diameter will actually appear as an X-axis amount in the program. A disadvantage is that the actual depth still has to be calculated and a proper grooving tool selected. Example in Figure 36-5b does show the groove depth but, most likely, the bottom diameter will have to be calculated manually. Both dimensioning examples are about equally common in CNC programming.

Deep grooves are usually grooves that have a reasonably normal width but also have a much deeper ratio between the groove top diameter and its bottom diameter.

E. SIMPLE GROOVE PROGRAMMING

The simplest of all grooves is the one that has the same width and shape as the tool cutting edge - Figure 36-6.

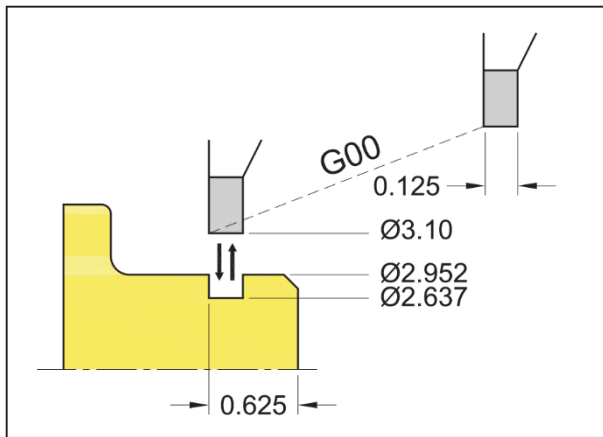


Figure 36-6

Simple groove example - program O3601

Insert width is equal to the groove width

To program such a groove is very straightforward. In rapid mode, move the grooving tool to its starting position, feed-in to the groove depth, then rapid out back to the starting position, and - the groove is finished. There are no corner breaks, no surface finish control, and no special techniques used. Some will say, and no quality either. A dwell at the groove bottom may be the only improvement. True, quality of such groove will not be the greatest, but a groove it will be. Groove of this type is strictly a utility groove and is hardly ever required in precision manufacturing. At the same time, programming such grooves is a good start to learn more advanced techniques.

The following program O3601 uses standard 0.125-inch square grooving insert, for a groove of the same width. Groove depth is the single difference between the two diameters given in the drawing:

$$(2.952 - 2.637) / 2 = 0.1575$$

The sample program uses tool T08, as the last tool:

O3601 (SIMPLE GROOVE) (G20) ...

N33 T0800 M42 (TOOL 8 ACTIVE)

N34 G97 S650 M03 (650 RPM SPEED)

N35 G00 X3.1 Z-0.625 T0808 M08 (START POINT)

N36 G01 X2.637 F0.003 (FEED-IN TO DEPTH)

N37 G04 X0.4 (DWELL AT THE BOTTOM)

N38 X3.1 F0.05 (RETRACT FROM GROOVE)

N39 G00 X6.0 Z3.0 T0800 M09 (CLEAR POSITION)

N40 M30 (END OF PROGRAM)

%

Program O3601 does the following. First, imperial mode of input is preset from the program beginning with G20. Blocks N33 and N34 are startup blocks for tool T08, with direct *r/min* selected. *Constant Surface Speed* (CSS) in G96 mode can be selected instead. N35 is a block where the tool moves to a position from which the groove will be started (start point). Clearance at this safe location is the clearance above the part diameter, which is 0.074 inches in the example:

$$(3.1 - 2.952) / 2 = 0.074$$

Coolant is applied in the same block, during tool motion. Block N36 contains the actual groove

plunging cut, at a slow feed rate of 0.003 in/rev. Block N37 is a dwell time of 0.4 seconds, followed by tool return to the starting diameter and completion of the program.

Although this particular grooving example was very simple, evaluate the program a little more. It contains several important principles that can be applied to the method of programming any groove, where its precision and surface finish are very critical.

Note the clearance *before* groove cutting begins. Tool is positioned 0.074 inches above the part diameter, at 3.100. Always keep this distance to a certain safe minimum. Grooves are usually cut at a slow feed rate, and it may take too much time just to cut in the air. Also note the actual cutting feed rate has increased from 0.003 in/rev in block N36 to a rather high feed rate of 0.050 in/rev in block N38. Rapid motion command G00 could have been used instead. Feeding out at a heavier feed rate (rather than using a rapid motion), may improve the groove surface finish by eliminating tool drag on the walls (sides).

Tool width of 0.125 never appears in the program, directly or indirectly. That means the insert shape and width will become the shape and width of the groove. It also means a different groove width, if another insert size is used, although the program structure remains unaffected. The structure will remain unaffected even if the grooving tool *shape* is changed. Combination of the shape *and* size change will offer endless opportunities, all of them being possible without a single change to the program.

F. PRECISION GROOVING TECHNIQUES

A simple *in-out* groove will not be good enough. Its sides may have a rough surface, outside corners will be sharp and its width is dependent on insert width and its wear. For most of machining jobs, such a groove is not acceptable.

To program and machine any precision groove requires extra effort, but the result will be a high-quality groove. This effort is not always justified, as high quality comes with a price. The next two illustrations show groove dimensions and program related details. Drawing in *Figure 36-7* shows a high precision groove, although its width is intentionally exaggerated for impact of the example.

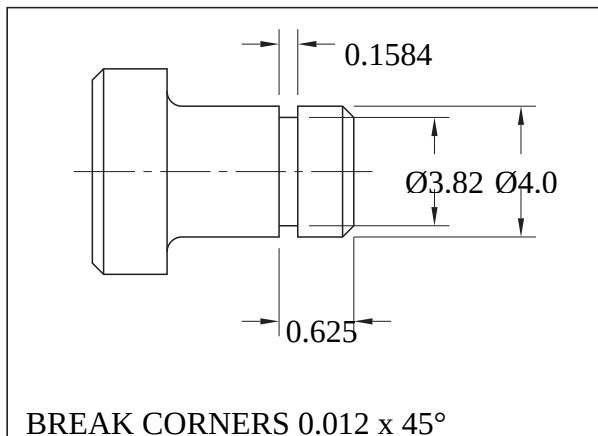


Figure 36-7

Drawing for a precision groove example O3602

What is the best cutting method? One plunge rough cut and two finish cuts, one for each wall, are reasonable; so is the 0.006 stock added to the bottom diameter. Also, sharp corners will be broken with a 0.012 chamfer at 4.0. *Figure 36-8* shows distribution of the cuts.

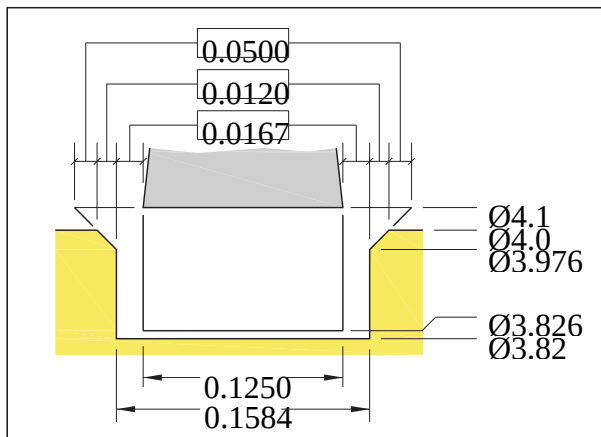


Figure 36-8

Precision groove - distribution of cuts for the example O3602

Before the first block can be programmed, selection of cutting tool and machining method is a sign of good planning. These are important decisions because they directly influence the final groove size and its condition.

Groove Width Selection

Grooving tool selected for the example in program O3602 will be an external tool, assigned to tool station number three - T03. Tool reference point is selected at the *left edge* of the insert, which is a standard selection. Insert width has to be selected as well. Grooving inserts are available in a variety of standard widths, usually with an increment of 1 mm for metric tools, and 1/32 or 1/16 inch for tools in imperial system. In this case, non-standard groove width is 0.1584 inch. The nearest standard insert width is 5/32 inch (0.15625 inch). The question is - should we select the 5/32 inch insert width? In a short answer, no. In theory, this insert *could* cut the groove, but because the actual difference between the insert width and the groove width is so small (0.00215 inch over two walls), there is very little material to cut.

Dimensional difference would allow only slightly more than 0.001 per each side of the groove, which may cause the insert to rub on the wall rather than cut it. A better choice is to step down to the next lower standard insert width, that is 1/8th of an inch (0.1250). There is much more flexibility with 1/8 width than with 5/32 width. Once the grooving tool is selected, initial settings can be assigned offset number (O3), spindle speed (400 ft/min), and gear range (M42) - and a note for the setup sheet:

T0303 = 0.1250 SQUARE GROOVING TOOL

The first few program blocks can now be written:

O3602 (PRECISION GROOVE) (G20) ...

N41 T0300 M42

N42 G96 S400 M03

...

Machining Method

Once the grooving tool has been selected and assigned a tool station number (tool turret position), the actual *method* of machining the groove has to be decided. Earlier, the machining method has been described generally, now a more detailed description is necessary.

One simple programming method is *not* an option - the basic *in-out* technique used earlier. That means a *better* method must be selected, a method that will guarantee a high-quality groove. The first step towards that goal is realization of the fact that a grooving insert with the width narrower than the groove width, will have to be plunged into the groove more than once. How many times?

It is not difficult to calculate that a groove 0.1584 wide and machined with a 0.1250 wide grooving insert, will need *at least* two grooving cuts. But what about a groove that is much wider than the groove in the example?

There is an easy way to calculate the *minimum number of grooving cuts* (or plunges), using the following formula:

$$C_{\min} = \frac{G}{T_w}$$

+ where ...

$C_{\min}$ = Minimum number of cuts G_w = Groove width for machining

T_w = Grooving insert width

Applying the formula to the example, the starting data are the groove width of 0.1584 of an inch and the grooving insert width of 0.1250 of an inch. That translates into the minimum of *two* grooving cuts. Always round upwards, to the nearest integer: $0.1584/0.1250=1.2672=2$ cuts.

A possible decision could be to plunge once to finish the left side of the groove and, with one more plunge, to finish the groove right side. The necessary overlap between the two cuts is guaranteed and the only remaining operation is the chamfering. A groove programmed this way may be acceptable but will *not* be of a very good quality.

Even if only an *acceptable* quality groove is produced during machining, such a result does not give the programmer much credit. What can be actually done to assure the *highest* groove quality possible?

In order to write first class programs, make the best efforts to deliver an exceptional quality at the *programming* level, in order to prevent problems at the *machining* level

How can this suggestion be used with the example? The main key is knowledge of machining processes. Machining experience confirms that removing an *equal* stock from each wall (side) of the groove will result in better cutting conditions, better surface finish control and better tool life.

If this observation is used in the current example, an important conclusion can be made. If two plunge cuts of uneven width will yield at least *acceptable* results, *three* cuts that are *equally distributed* should yield even better results.

If *at least three* grooving cuts are used to form the groove rather than the minimum two cuts, the CNC programmer will gain control of two always important factors:

Control of the groove POSITION

Control of the groove WIDTH

In precision grooving, these two factors are equally important and should be considered together.

Look carefully at how these factors are implemented in the example. The first factor applied under program control is the groove position. Groove *position* is given in the drawing as 0.625 inches from the front face of part to the groove left side. There is no plus or minus dimensional tolerance specified, so the drawing dimension is used as arbitrary and is programmed directly. The second factor under program control is the groove *width*. That is 0.1584 of an inch on the drawing and the selected tool insert width is 0.1250. The goal is to program cutting motions in *three steps*, using the technique already selected:

STEP 1

Rough plunge in the groove middle, leaving an equal material stock on both groove faces for finishing - also leave small stock on the groove bottom

STEP 2

Program grooving tool operation on the groove left side, including the chamfer (corner break)

STEP 3

Program grooving tool operation on the groove right side, including the chamfer (corner break) and sweep groove bottom towards the left wall

The last two steps require chamfer cutting or a corner break. Width of the chamfer *plus* the width of any subsequent cut should never be larger than about one half to three quarters of the insert width. In the third step, sweeping of groove bottom is desired. That suggests the need to consider stock allowances for finishing.

Finishing Allowances

During the initial step, the first plunge has to take place at the *exact* center of the groove. To calculate the Z-axis position for the start, first find the amount of stock on each wall that is left for finishing. The stock amount will be one half of the groove width minus the insert width - see details in the previous *Figure 36-8*:

$$(0.1584 - 0.1250) / 2 = 0.0167$$

The tool Z-position will be 0.0167 on the positive side of the left wall. If this wall is at Z-0.625, the grooving tool start position will be at Z-0.6083. When the tool completes the first plunge, there will be an equal amount of material left for finishing on both walls of the groove.

Do your best to avoid rounding off the figure 0.0167, for example, to 0.0170 inch. It would make no difference for the machining, but it is a sound programming practice to use *only* calculated values. The benefit of such approach is in eventual program checking, and also with general consistency in programming. Equal stock amounts offer this consistency; 0.0167 and 0.0167 is a better choice than 0.0170 and 0.0164, although the practical results will be the same.

Next look is at the X-axis positions. The first position is where the plunge will start from, the second position is the end diameter for plunging cut. A good position for the start is about 0.050 *per side* above the finished diameter, which in this case would be a clearance diameter calculated from the 4.0:

$$4.0 + 0.05 \times 2 = 4.1 \quad (\text{X4.1})$$

Do not start the cut with a clearance of more than 0.050 inch (1.27 mm) - with slow feed rates that are typical to grooves, there will be too much air to cut, which is not very efficient. End diameter is the groove bottom, given on the drawing as 3.82. Dimension of X3.82 could be programmed as the target diameter, but it does help to leave a very small stock, such as 0.003 per side (0.006 on diameter), to make a *sweep finish* of the groove bottom. That will add two times 0.003 to the 3.82 groove diameter, for the programmed X target as X3.826. Once the plunge is done, tool returns to the start diameter:

N43 G00 X4.1 Z-0.6083 T0303 M08

N44 G01 X3.826 F0.004 N45 G00 X4.1

Rapid motion back above the groove (N45) is a good choice in this case, because the sides will be machined later with the finishing cuts, so the surface finish of the walls is not critical at this moment. After roughing the groove, it is time to start the finishing operations.

All calculated amounts can be added to the previous *Figure 36-8*, and create data for a new *Figure 36-9*:

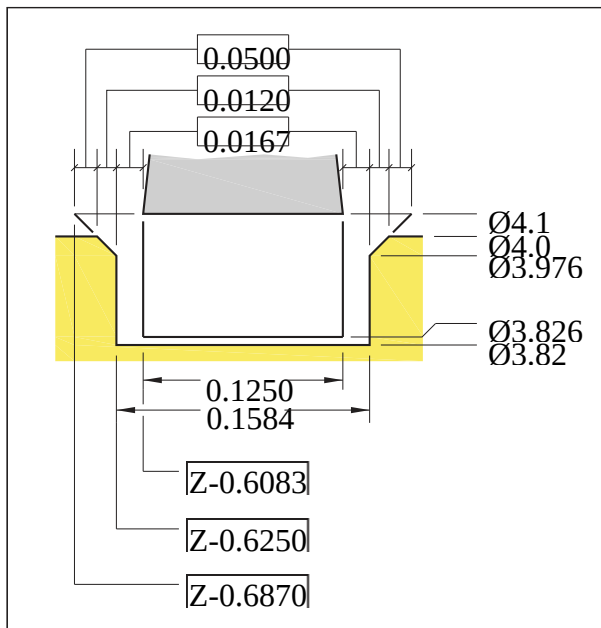


Figure 36-9

Precision groove - groove data used in program O3602

Groove Tolerances

As in any machining, program for grooves must be structured in such a way, that maintaining tolerances at the machine will be possible. There is no specified tolerance in the example, but it is implied as very close by the four-decimal place dimension. A tolerance range, such as 0.0 to +0.001, is probably a more common way of specifying a tolerance. Only the dimensional value that falls *within* the specified range can be used in a program. In this example, the aim is the drawing dimension of 0.1584 (selected intentionally).

A possible problem often encountered during machining and a problem that influences the groove width the most, is a *tool wear*. As the insert works harder and harder, it wears off at its edges and actually becomes *narrower*. Its cutting capabilities are not necessarily impaired, but the resulting groove width may not fall within close tolerances. Another cause for an unacceptable groove width is the *insert width*. Inserts are manufactured within high level of accuracy, but also within certain tolerances. If an insert is changed, the groove width may change slightly, because the new insert may not have exactly the same width as the previous one. To eliminate, or at least minimize, the possible *out of tolerance* problem, use quite a simple technique - program an *additional* offset for finishing operations only.

Earlier, when the precision groove was planned, offset 03 had been assigned to the grooving tool. Why would an *additional* offset be needed at all? Assume for a moment, that all machine settings use just a single offset in the program. Suddenly, during machining, the groove gets narrower due to tool wear. What can be done? Change the insert? Modify the program? Change the offset? If the Z-axis offset setting is adjusted, either to the negative or positive direction, that will change the groove *position* relative to program zero but it will *not change the groove width!* What is needed is a second offset, an offset that controls the groove width only. In program O3602, the left chamfer and side will be finished with one offset (03), the right chamfer and side will use a second offset. To make the second offset easier to remember, number 13 will be used.

One other step has to be finished first - calculation of the left chamfer *start* position. Currently, the tool is at Z-0.6083 but has to move by the wall stock of 0.0167 and the chamfer width 0.012 as well as clearance of 0.050 - for a total travel of 0.0787, to Z-0.687 position. At a slow feed rate, the chamfer is done first and the cut continues to finish the left side, *to the same diameter* as for roughing, which is X3.826:

N46 Z-0.687

N47 G01 X3.976 Z-0.625 F0.002 N48 X3.826 F0.003

The next step is to return the tool above part diameter. This motion is more important than it seems. In the program, make sure the finished left side is not damaged when the tool retracts from the groove bottom. Also make sure the grooving tool will not contact the right side wall stock. That means do not retract the tool further then the position of Z-0.6083. It also means do *not* rapid out, because of a possible contact during the 'dogleg' or 'hockey stick' motion, described on *page 149*. The best approach is to return to the initial start position at a relatively *high but non-cutting* feed rate:

N49 X4.1 Z-0.6083 F0.04

At this point, the left side wall is finished. To program similar motions for the right side wall, the tool has to cut with its *right* side (right edge). One method is to change the G50 coordinates in the program, if this older setting is still used, or use a different work coordinate offset. The method used here is probably the simplest and also the safest. All motions relating to the right chamfer and right side groove wall will be programmed in *incremental mode*, applied to the Z-axis only, using the W address:

N50 W0.0787 T0313

N51 X3.976 W-0.062 F0.002

In block N50, the tool travels the total distance equivalent to the sum of right wall stock 0.0167, chamfer of 0.012 and clearance of 0.050. In the same block, the second offset is programmed. This is the only block where offset 13 *should* be applied - one block before, it's too early, and one block, after it's too late.

Block N51 contains the target chamfer position and absolute mode for X-axis and is combined with incremental mode for Z-axis.

To complete the groove right side wall, finish the cut at the full bottom diameter, block N52, then continue to remove the stock of 0.003 from bottom diameter (block N53) - this is called *sweeping the groove bottom*:

N52 X3.82 F0.003

N53 Z-0.6247 T0303

Also look at the Z-axis end amount - it is a small value that is 0.0003 short of the 0.625 drawing dimension! The purpose here is to compensate for a possible tool pressure. *There will not be a step in the groove corner!* Because the sweep will end at the groove *left* side, the original offset (03) must be reinstated. Again, block N53 is the only block where the offset change is correct. Make sure not to change tool numbers - *the turret will index !*

The intended program O3602 can now be completed. All that remains to be done is return to the groove starting position, followed by program termination blocks:

N54 X4.1 Z-0.6083 F0.04

N55 G00 X10.0 Z2.0 T0300 M09

N56 M30 %

At this point, *complete* program O3602 can be written. Note program blocks where the offset has been changed, they are identified in the comment section:

O3602 (PRECISION GROOVE)

20) ...

N41 T0300 M42

(NO

(FSET)

N42 G96 S400 M03

N43 G00 X4.1 Z-0.6083 T0303(OFFSET
08)

N44 G01 X3.826 F0.004

N45 G00 X4.1

N46 Z-0.687

N47 G01 X3.976 Z-0.625 F0.002

N48 X3.826 F0.003

N49 X4.1 Z-0.6083 F0.04

N50 W0.0787 T0313 (OFFSET
)

N51 X3.976 W-0.062 F0.002

N52 X3.82 F0.003

N53 Z-0.6247-T0303 (OFFSET 03)

N54 X4.1 Z-0.6083-F0.04

N55 G00 X6.0 Z3.0 T0300 M09 (NO OFFSET)

N56 M30

%

WARNING !

It is very important to use caution when a double tool offset for a single tool is used during machining

(This warning applies generally - not only for grooving)

Remember that the *purpose* of offset in the example is to control the groove *width*, *not* its diameter.

Always follow these precautions, based on the example program O3602:

Start machining with identical initial amounts assigned to both offsets (the same XZ values for offsets 03 and 13).

X-offset amounts of 03 and 13 must always be the same. If the X-setting of one offset is changed, setting of the other offset must be changed to the same amount. Adjust both X-offsets to control the groove depth tolerance

If the groove width becomes too narrow and has to be adjusted, only the Z-offset amount is changed

To adjust the groove left side wall position, change the Z-offset number 03

To adjust the groove right wall position, change the Z-offset number 13

Do not cancel the current offset - change from one to the other offset directly

Make sure the tool number (the first two digits of the T address) does not change, otherwise, *THERE WILL BE A TOOL CHANGE!*

Other precautions can be added, depending on the exact conditions. Use common sense, and

always check the program carefully, before it is released to production. If a tool number is changed in the program, ***change all T-words!***

Groove Surface Finish

Programming just about any precision groove should be fairly easy from now on. Only a few last notes on the subject of groove cutting as they relate to surface finish. Just by following the suggested methods of equal cut distribution, proper spindle speeds and feed rates, good condition of the cutting tool and insert, suitable coolant, and other techniques used in the example, surface finish will almost take care of itself.

Keep in mind, that the term '*precision groove*' does not only describe the precise groove position and its precise dimensions, but it also means a high-quality look, a look that often means much more than just a cosmetic feature.

G. MULTIPLE GROOVES

Multiple grooving is a common term used for cutting the same groove at different positions of the same part. In these cases, the program will most likely benefit from developing a subprogram (subroutine) for multiple grooves, that will be called at various groove locations. Subprograms save valuable programming time, they are easily designed and easily edited. Although subprograms will be discussed in a separate chapter (*see page 383*), an example of a multiple groove programming using a subprogram is shown at the end of this chapter, at least for reference and basic introduction.

When cutting multiple grooves, more material will be removed. On external diameter grooves, there are no special considerations necessary, gravity will take care of the extra chips. This is not the same situation for internal grooves. The moment several grooves are machined internally, there is a small pile of cutting chips accumulated in the bored hole. These chips can be in the way of a smooth cutting operation and could damage the bored diameter and even the grooving tool itself. To solve this problem, consider machining of only a few grooves, move the tool out and blow out chips from the internal area. Using the optional stop M01 can be useful in this case. When all chips have been removed, continue with the same tool to cut more grooves.

H. FACE GROOVES

Face grooving (sometimes incorrectly called *trepanning*) is a horizontal groove cutting process, with the tool moving along the Z-axis. Tool is programmed along the same principles as vertical grooving along the X-axis. Because of the nature of such a grooving cut, tool *orientation* presents the most important single consideration in face grooving. Main issue is the *radial clearance* of the cutting insert, during a cut. There is no need to worry too much about radial clearance for vertical grooving, because the insert cutting edge is on the same plane as machine center line. However, in horizontal grooving, the insert clearance along the cut radius is of utmost importance.

The next example shows how to program a typical face groove and is illustrated in *Figure 36-10*.

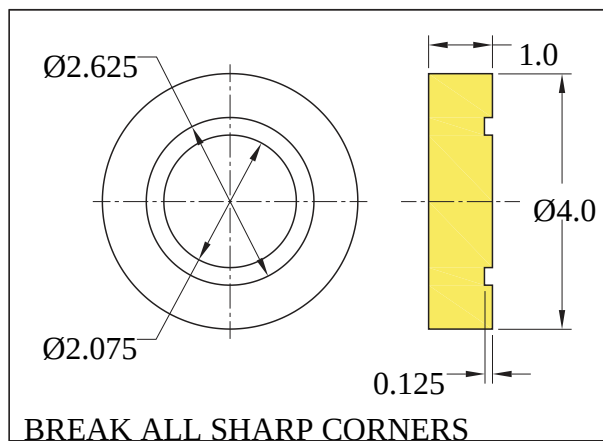


Figure 36-10

Face grooving example - program O3603

Although both external and internal groove diameters are engineering choices in the drawing, physical groove width is necessary for programming as well. To calculate the groove width, use a simple calculation - find one half of the difference between the two grooving diameters - that is:

$$(2.625 - 2.075) / 2 = 0.275$$

This is the actual groove width amount, 0.275 in the given example. Always keep in mind that the program will use a smaller 0.250 wide face grooving insert. Following the programming examples of precision groove listed earlier, three cuts will be made - one rough plunge in the middle of the groove, and two finishing cuts, with a small corner break. But first, let's look at the *radial clearance* of the grooving tool. This is a very important programming consideration; one that is unique to most face grooving operations, yet it is also one that is easy to be overlooked.

Radial Clearance

Many grooving inserts are relatively high, in order to give them strength. Grooving insert for face grooving operations is mounted at 90 towards the part face (parallel to the spindle center line). A standard grooving insert has virtually no radial clearance and most likely will interfere with the part at its lower end - *Figure 36-11*.

From the illustration is obvious that the grooving insert cannot be used as is and has to be modified. Such a modification is usually done by grinding a suitable radial clearance, as illustrated in *Figure 36-12*.

This is a simple operation, providing the proper grinding tools are available. Make sure that the grinding does not affect the insert width and only minimum necessary material is removed, otherwise the tool loses strength.

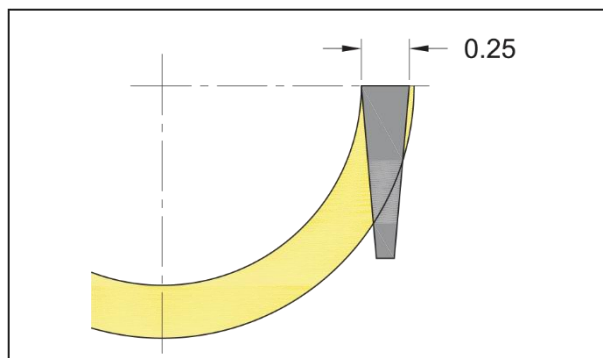


Figure 36-11

Interference of a standard grooving insert on a face groove

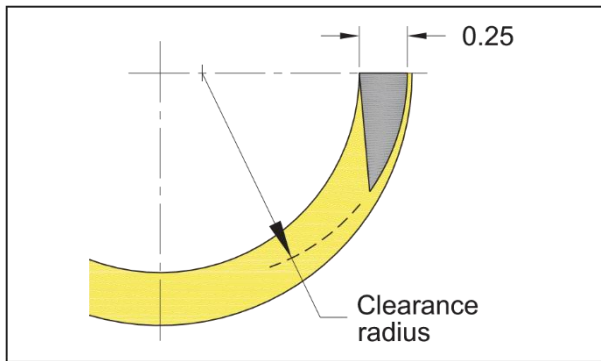


Figure 36-12

Standard grooving insert modified for face grooving

Face Grooving Program Example

Program O3603 uses modified insert and a 0.012 corner break, to eliminate sharp corners. Only one offset is used in the program. The tool set point is the lower edge of insert, corresponding to the 2.075. All calculations should be easy to retrace, they use exactly the same procedures as those described for vertical grooving:

O3603 (FACE GROOVE) (G20) ...

N21 T0400 M42

N22 G96 S450 M03

N23 G00 X2.1 Z0.05 T0404 M08

N24 G01 Z-0.123 F0.003

N25 Z0.05 F0.04

N26 X1.951

N27 X2.075 Z-0.012 F0.001

N28 Z-0.123 F0.003

N29 X2.1 Z0.05 F0.04

N30 U0.149

N31 U-0.124 Z-0.012 F0.001

N32 Z-0.125 F0.003

N33 X2.0755

N34 X2.1 Z0.05 F0.04 M09

N35 G00 X8.0 Z3.0 T0400

N36 M30 %

I. CORNER GROOVES / NECK GROOVES

Corner grooving is also a grooving operation, one that uses a special grooving insert designed to cut along a 45 angular motion. Resulting groove can be square or with a radius, depending on the tool and insert used and design required. Grooving insert may also be a standard type of insert, placed into a 45 angle tool holder. The purpose of this type of a groove is to allow machining of recesses and undercuts, usually in a corner of the part. It assures a shoulder match of two assembled components.

To program a corner groove (neck groove), the radius of the grooving insert must be known,

0.031 (1/32) of an inch in the example. Cutting depth is established from the drawing data. Normally, the corner groove is specified as a 'minimum undercut'. In this case, the center of undercut will be at the intersection of the shoulder and diameter. Cutting motion in and out of the groove must be at 45, meaning an identical amount of travel in both X and Z axes. *Figure 36-13* illustrates a corner groove with a 0.031 radius minimum undercut.

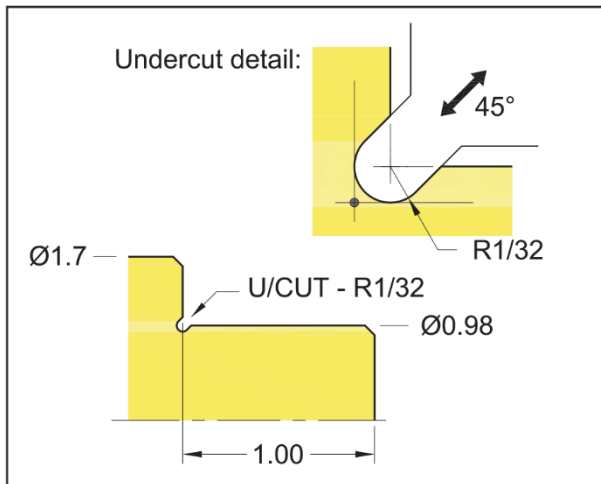


Figure 36-13

Corner groove - undercut program example O3604

The program itself has no hidden surprises and is not difficult to complete or interpret:

O3604 (CORNER GROOVE)
20) ...

N217 G50 S1000 T0500 M42
18 G96 S375 M03

N219 X1.08 Z-0.95 T0505 M08

N220 G01 X0.918 Z-1.031 F0.004 (CUT IN)

N221 G04 X0.1 (DWELL)

N222 X1.08 Z-0.95 F0.04 (MOVE

N223 G00 X6.0 Z3.0 T0500 M09 JT)

N224 M30

%

Block N219 positions the tool in such a way that the insert center (as well as the setup point) is in on center line of the neck groove (0.050 clearance in X and Z axes). Blocks N220 and N222 are the two cutting motions - one into the groove in N220, the other out of the groove in N222. The amount of travel is exactly the same in either direction. dwell of 0.1 second is added for convenience at the neck bottom. Block N220 can also be programmed as an *incremental* motion for easier visualization:

N220 G01 U-0.162 W-0.081 F0.004

N221 G04 X0.1 N222 U0.162 W0.081 F0.04

J. GROOVING CYCLES

Fanuc controls for lathes offer two multiple repetitive cycles G74 and G75 that can be used for an interrupted cutting along an axis. Programming formats for both cycles were described in the previous chapter (*page 332*). G74 cycle is used for cutting along the Z-axis and is used mostly for

peck drilling, G75 is used for cutting in the X-axis, and is used mostly for simple grooving.

G75 Cycle Applications

Although used mainly for grooving, G75 cycle can also be used for an interrupted cut in facing. This cycle is quite simplistic to be of any use for high quality surface finish, but it does have its benefits. Its main purpose is to break chips while cutting along the X-axis. This may be useful for some grooving and part-off operations, as well as face cutting. Another use is for roughing the core out of deep grooves, so they can be finished later by more precise methods.

In G75, the chip breaking is done by alternating between a cutting motion in one direction and a rapid retract motion in the opposite direction. This means that one cutting motion is always followed by an opposite rapid motion, on the basis of *feed-in-rapid-out* principle and a built-in clearance. *Figure 36-14* illustrates the concept.

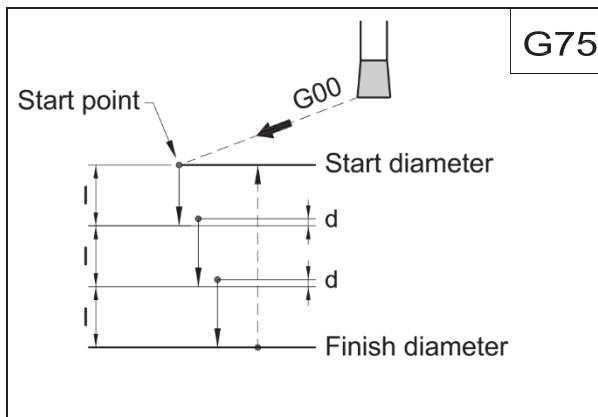


Figure 36-14

Schematic representation of the G75 cycle

Figure 36-15

Single groove example using the G75 cycle - program O3605

The following program example O3605 cuts a single groove and is based on *Figure 36-15*.

O3605

(G75 SINGLE GROOVE) (G20) ...

N43 G50 S1250 T0300 M42

N44 G96 S375 M03

N45 G00 X1.05 Z-0.175 T0303 M08

N46 G75 X0.5 I0.055 F0.004

N47 G00 X6.0 Z2.0 T0300 M09

N48 M30

%

Note that the I amount is 0.055. This is not a value without a meaning. In fact, it is a carefully calculated depth of each groove peck. The tool travel will be from 1.050 to 0.500, or 0.275 per side $(1.05-0.50)/2=0.275$. There will be exactly five grooving pecks of 0.055 each $(0.275/5=0.055)$.

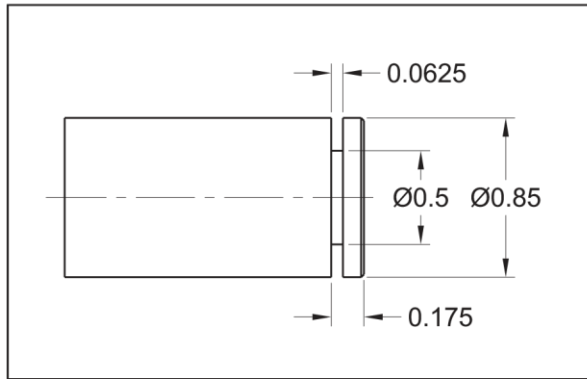
Multiple Grooves with G75

It is possible to program multiple grooves very easily, using G75 cycle. In this case, the groove spacing, the pitch between grooves must always be equal, otherwise G75 cycle cannot be used. The clearance specification *d* in *Figure 36-14* is normally not programmed.

Motion retract amount is built within the cycle and is set by an internal parameter of the control system. In *Figure 36-14* it is identified by the value *d*, (usually set to approximately 0.010 to 0.020 inches in the control). The next two examples illustrate the practical use of G75 grooving cycle.

◆ Single Groove with G75

A single groove requires the X and Z coordinate of the starting point, the final groove diameter X, and the depth of each cut I. For a single groove, the Z-axis position and the K distance cannot be programmed. The Z position is given by the starting point and does not change.



N86 G00 X6.0 Z2.0 T0300 M09

N87 M30

%

Setup and conditions for multiple grooves are identical to those for a single groove. The only difference is additional entries in the G75 cycle call.

This technique may be used not only for multiple grooves separated by solid material, but also for opening up a single groove that is much wider than the grooving insert. The only difference in programming will be the amount of K the distance between grooves. If K is greater than the insert width, several individual grooves will be cut. If K is equal to or smaller than the insert width, a single wide groove will be cut. Experiment to find the best amounts.

K. SPECIAL GROOVES

There are many more types of grooves than can be described in this digital book. They are mainly grooves of special shapes, used by specific industries - grooves that serve a certain purpose. The most typical grooves of this type are round grooves, pulley grooves, O-ring grooves, and several others. Certain grooves, usually those that conform to common industrial standards, can be machined with readily available inserts. A typical example of this kind of grooving is a pulley groove. The programming principles for 'nonstandard' grooves are no different than those described in this chapter.

L. GROOVES AND SUBPROGRAMS

Programming multiple grooves with G75 cycle is usually not the preferred method for precision work. The two main drawbacks are the groove quality and the requirement for an equal spacing between grooves. There is another method to program multiple grooves - much more efficient method - one that uses subprograms.

Multiple grooves can be programmed very efficiently and with much increased precision by using the technique of subprograms, a subject starting on *page 383*. The guiding principle is to program all common groove motions in the subprogram and all motions that vary from groove to groove, in the main program. This way, the same groove can be repeated at fixed intervals or

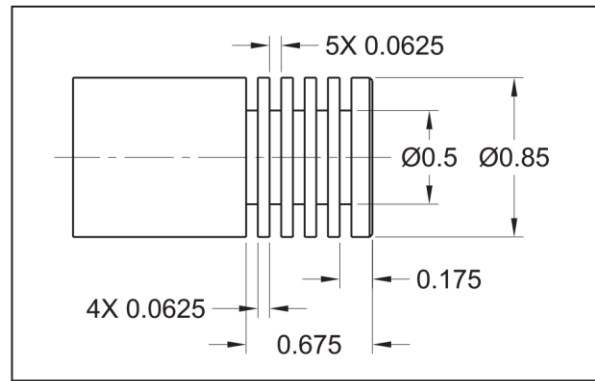


Figure 36-16

Multiple groove example using the G75 cycle - program O3606

Program example O3606 for multiple grooves, using G75 cycle, is based on *Figure 36-16*.

```
O3606
(G75 MULTIPLE GROOVES)
(G20)
...
N82 G50 S1250 T0300 M42
N83 G96 S375 M03
N84 G00 X1.05 Z-0.175 T0303 M08
N85 G75 X0.5 Z-0.675 I0.055 K0.125 F0.004
```

variable intervals, as needed.

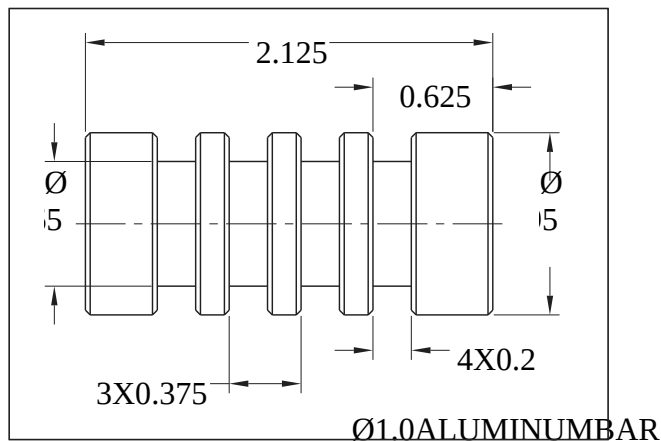


Figure 36-17

Multiple grooves programming using a subprogram O3607 is the main program and O3657 is the subprogram

In Figure 36-17 is a simple example of a multiple groove programming, using a subprogram. Only two cutting tools are used - a turning tool used for facing and turning and a 0.125 wide part-off tool that machines the four grooves, cuts the back chamfer and parts-off the finished part. Part-off operations are described in the next chapter. Note that all tool motions related to the groove position are programmed in the main program O3607, all tool motions related to the actual groove cutting are programmed in subprogram O3657. Equal spacing between grooves is used for the example.

O3607 (GRV W/SUB-PROG)

(T01 - 55 DEGREE DIAMOND INSERT)

N1 G20 T0100

N2 G96 S500 M03

N3 G00 X1.2 Z0 T0101 M08

N4 G01 X-0.07 F0.006 (FACE OFF)

N5 G00 Z0.1

N6 G42 X0.7 (START OF CHAMFER)

N7 G01 X0.95 Z-0.025 F0.003 (CHAMFER)

N8 Z-2.285 (TURN OD)

N9 U0.2 F0.03

N10 G00 G40 X4.0 Z4.0 T0100

99 N11 M01

(T05 - 0.125 PART-OFF
TOOL)

N12 G50 S2500 T0500

N13 G96 S500 M03

N14 G00 Z-0.5875 T0505 M08(POS-GRV1)
5 X1.0

N16 M98 P3657 (CUT GRV

N17 G00 W-0.375 M98 P3657 (CUT GRV
N18 G00 W-0.375 M98 P3657 (CUT GRV
N19 G00 W-0.375 M98 P3657 (CUT GRV
N20 G00 Z-2.285 (OPEN UP FOR PART-OFF)
N21 G01 X0.8 F0.006
N22 X1.1
N23 G00 X1.0 Z-2.2 (CHAMFER BACK START) N24 G01 X0.9 Z-2.25 F0.003
(CHAMFER)
N25 X-0.02 F0.005 (PART-OFF)
N26 G00 X1.2 (CLEAR)
N27 G40 X4.0 Z4.0 T0500 M09
N28 M30
%
O3657 (SUB-PROG FOR O3607)
N101 G01 X0.66 F0.004 (FEED TO ROUGH OD)
N102 G00 X1.0 (CLEAR OUT)
N103 W-0.0875 (SHIFT TO LEFT CHFR) N104 G01 X0.9 W0.05 F0.002 (LEFT CHFR)
N105 X0.66 F0.004 (FEED TO ROUGH OD)
N106 X1.0 W0.0375 F0.03 (BACK TO START)
N107 W0.0875 (SHIFT TO RIGHT CHFR)
N108 X0.9 W-0.05 F0.002 (RIGHT CHFR)
N109 X0.65 F0.004 (FEED TO FINISH OD)
N110 W-0.075 (SWEEP BOTTOM)
N111 X1.0 W0.0375 F0.03 (BACK TO START)
N112 M99 (RETURN TO MAIN)
%

This example completes the chapter related to grooving. Although grooving is a relatively simple machining operation, programming grooves can become a significant challenge in certain cases.

29. PART-OFF

Part-off, sometimes called a *cutoff*, is a machining operation typical to lathe work, usually using a bar feeder attachment. During a part-off, the cutting tool (or part-off tool) separates the completed part from the bar stock. The completed part will fall off the bar, usually into a special bin to protect it from damage. Part catcher may also be used.

A. PART-OFF PROCEDURE

Programming procedure for a part-off toolpath is very similar to the grooving procedure. In fact, part-off is an extension of grooving. The purpose of part-off is somewhat different, because its objective is to separate the completed part from the stock material, rather than create a groove of certain width, depth and quality. Material bar stock is usually a long round/hex rod that is 8, 10, 12 or more feet long.

Two most important considerations in part-off are the same as those for standard grooving. One is the chip control, the other is coolant application.

Parting Tool Description

Part-off uses a special cutting tool. Such a tool used for part-off is called a parting tool or a part-off tool. Sometimes the term *cutoff* is used for this kind of a tool, as well as the machining method; it has the same meaning as the term *part-off*. Part-off tool is similar in design to a grooving tool, with one major difference. The *length* of the cutting blade is much longer than that of a grooving tool, making it suitable for deep grooves. A typical example of a part-off tool is illustrated in Figure 37-1.

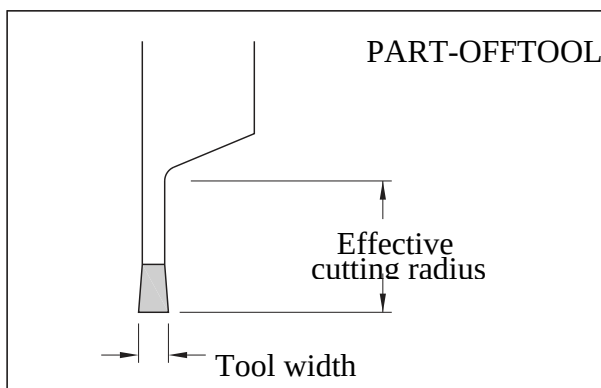


Figure 37-1

Part-off tool - cutting end configuration

At the end of the metal blade is usually a carbide insert, with clearance angles on both sides. The tool cutting end is available in several different configurations, always at the end tip of the carbide portion. The most typical tool end configurations are shown in the following illustration Figure 37-2:



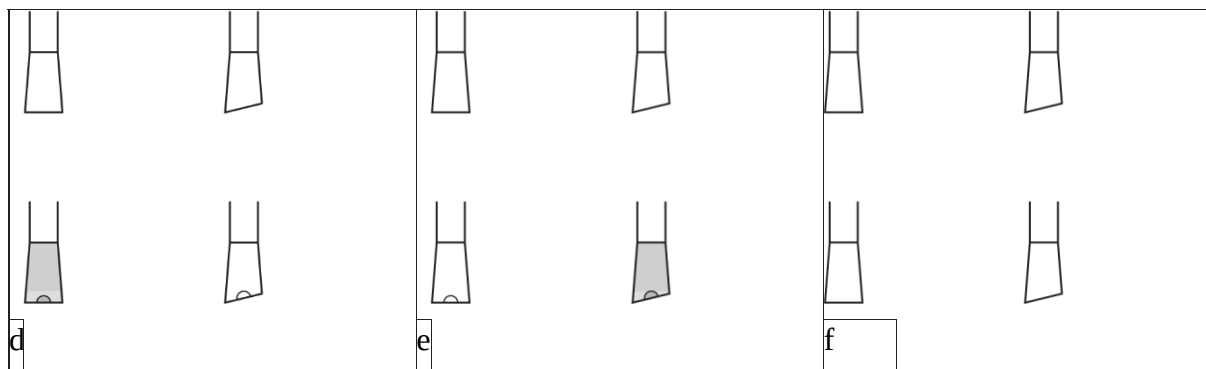


Figure 37-2

Part-off tool - cutting tip configurations

Note the two kinds of each grooving insert design shown - the series without a dimple (items *a*, *b* and *c*), and the series with a dimple (items *d*, *e* and *f*). The dimple is an intentional dent pressed in the middle of the cutting edge that deforms the chip and helps in coiling it. The result is a chip that is narrower than the width of cut. Such a chip does not clog the generated groove and extends tool life, although it may cost a little more.

Also note a slight angle on *b*, *c*, *e* and *f* styles. This angle helps in controlling the size and shape of the stub left on the part when it is separated from solid bar. It also controls the rim size that is left over on the part when parting-off a tubular bar. Although all designs have their special applications, probably the most versatile choice would be the style *f*, particularly for large cutting diameters. Unlike in the other types of machining, the cutting chips for part-off should *coil*, not break. The cutting insert with a dimple or a similar design is best suited for that purpose.

It is a common practice amongst programmers to use only one parting tool for all the work. They select the parting tool long enough to accommodate the maximum bar diameter and leave it permanently mounted in a tool holder, even for small diameters. The reasoning for this approach is that it saves a setup time. That is true to some extent but has a downside as well. A long part-off tool usually has a

wider insert than a short tool, in order to compensate for strength and rigidity. When the bar size is large, a long part-off tool is necessary, with its relatively wide insert. If such a tool is used for short parts, such as rings or other tubular stock with thin walls, it is the wrong tool selection that also wastes material. A short part-off tool with a narrower insert will justify the setup change.

A generous supply of coolant should always be made available at the cutting edge, just like for grooving. A water soluble coolant is a good choice, since it offers both the cooling and lubricating qualities. A typical mixture would be one part of soluble oil for 15-20 parts of water or as recommended by the coolant manufacturer. Make sure the coolant is supplied at full pressure, particularly for larger diameters. High pressure helps the coolant to reach the cutting edge and flush off chips that may accumulate in the opening.

Tool Approach Motion

The first step to program a part-off tool path is to select a part-off tool that has enough capacity to completely separate the part from a solid bar. The next decision is to select the insert width and location of the tool reference point. A part-off tool that is too short will not reach the spindle centerline safely. A tool too long may not be rigid enough, may cause vibrations, even break during the cut. The insert width is also important for good cutting conditions. Width of the tool is directly proportionate to the cutting depth capacity.

Selection of the part-off tool reference point follows the same rules as those for a grooving tool. It is definitely advantageous to have the same side for both, grooving and parting, to maintain setup consistency. The following programs illustrate the difference between the tool reference point on the *left* and on the *right* side of the tool tip - Figure 37-3 for program O3701 and Figure

37-4 for program O3702. In both cases, program zero is the front face of the finished part.

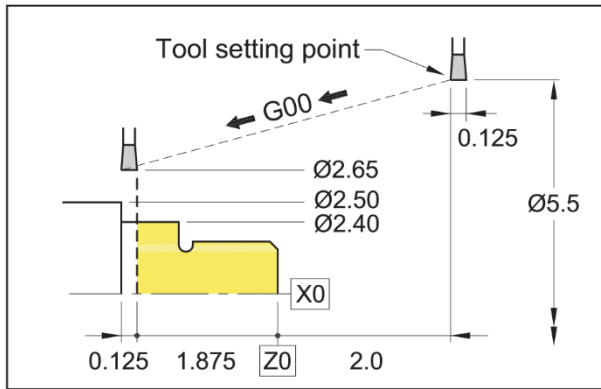


Figure 37-3

Part-off tool approach - left side tool reference - program O3701

Chapter 37

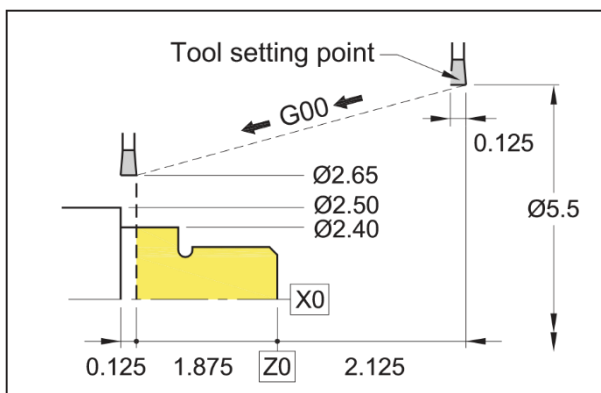


Figure 37-4

Part-off tool approach - right side tool reference - program O3702

In both examples, the tool change position and the final results are identical. Comparison of both programs shows dimensions of X-axis unchanged, but dimensions for the Z-axis are different (blocks N122 and N125). This reflects the cutting side of the tool tip.

O3701 (PART-OFF / LEFT SIDE TOOL EDGE) ...

N120 G50 S1250 T0800 M42

N121 G96 S350 M03

N122 G00 X2.65 Z-2.0 T0808 M08

N123 G01 X-0.03 F0.004

N124 G00 X2.65 M09

N125 X5.5 Z2.0 T0800

N126 M30

%

This example is consistent with the previous suggestions for precision grooving. Setting up the tool reference point on the *left* side of the tool is easier at the machine. If there is a good reason, set the tool reference point on the right side and program according to the *Figure 37-4*.

O3702 (PART-OFF / RIGHT SIDE TOOL EDGE) ...

N120 G50 S1250 T0800 M42 N121 G96 S350 M03

N122 G00 X2.65 Z-1.875 T0808 M08

—————
N123 G01 X-0.03 F0.004

N124 G00 X2.65 M09

N125 X5.5 Z2.125 T0800

N126 M30

%

One weakness of the left side tool setting is that the insert width has to be always added to the Z position in the program. In the second example, the final length of the part is used directly, but a possible collision with the chuck or collet is strong. Take care selecting the X-axis tool approach position and program the tool *above* stock diameter, even if the previous turning operations removed most of the stock. *Figure 37-5* shows correct and incorrect approaches of a part-off tool.

PART-OFF

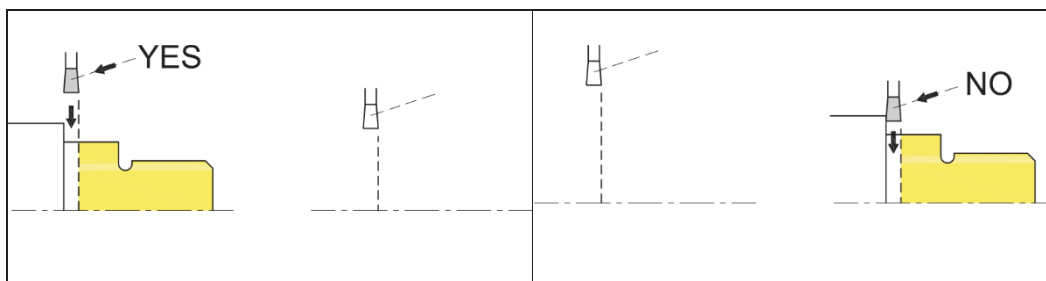


Figure 37-5

Correct and incorrect approach to stock diameter

Stock Allowance

Part-off operation does not always mean all machining has been completed. Often, part-off may complete only the first operation and additional machining will be necessary on the machined part. In such an event, some extra material (stock) has to be left on the back face, for subsequent finishing. Leave a stock amount of about 0.010 to 0.020 inches (0.3 to 0.5 mm). In that case, the block N122 would be changed in both programs - for example, from Z-2.0 to Z-2.015 in the first program example O3701 and from Z-1.875 to Z-1.89 in the second program example O3702.

Another program entry that is important to look at is the programmed X amount in block N122 - it is X2.65 in the example. That will leave 0.125 inches actual clearance above the 2.4 cutting diameter. If that seems a little too much, think again. Always consider the *actual* bar stock diameter, for safety reasons. In the example, the bar stock diameter is 2.500 inches, and the actual clearance will be a more reasonable 0.075 of an inch per side of the stock.

Tool Return Motion

Another safety aspect of programming a part-off tool is the method of returning to the tool change position, when parting operation is completed. It may be very tempting to replace the two program blocks N124 and N125 with a single block, then return to the tool change position immediately after the part-off - for example:

N124 G00 X5.5 Z2.0 (or Z2.125) T0800 M09

After all, the part has just been separated, fallen into the bin and one block in the program can be saved. *Don't do this*, it could become a very hazardous procedure. The part *should* have been removed by the tool and it *should* have fallen into the bin - but has all this actually happened? A variety of reasons may cause an *incomplete* part-off. The result is a broken tool, scrapped part,

possibly a damage to the machine itself.

Always return in the X-axis first, and always above the bar stock diameter

Part-off with a Chamfer

Not always the machined part will be done during a secondary operation. When the machining has to be completed with a part-off tool, it will require the best quality overall finish possible. One requirement of a good surface finish is broken sharp corners. In the example, a sharp corner is at the intersection of X2.4 and Z-1.875. If the turning tool cannot cut the chamfer during turning operation, part-off tool can be a good choice. Most part-off tools are not designed for cutting sideways (along the Z-axis), but chamfering removes only a small amount of material that is within the tool capabilities. Avoid chamfers that are wider than about 75% of the insert width or take several cuts if needed. The chamfer has to be cut *before* part-off and it should be cut from outside in, *not* from inside out. Correct programming technique for machining a chamfer during part-off is summed up in the following steps:

Position the tool further in the Z-axis than would be normal for regular part-off

Start the part-off operation and terminate it just below the diameter where the chamfer will end

Return to the starting diameter and move to the chamfer start position

Cut the chamfer in one block and part-off in the subsequent block

To illustrate this programming technique, study the following program example O3703 and illustration shown in *Figure 37-6* - tool reference point is on the left side, and the required chamfer is 0.020 inches at 45°:

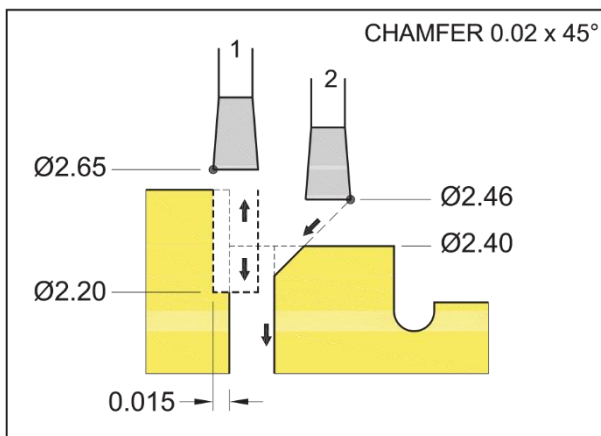


Figure 37-6

Corner breaking with a part-off tool - example O3703

O3703 (PART-OFF CHFR) (G20) ...

N120 G50 S1250 T0800 M42 N121 G96 S350 M03

N122 G00 X2.65 Z-2.015 T0808 M08

N123 G01 X2.2 F0.004

N124 X2.46 F0.03

N125 Z-1.95 (LEFT SIDE OF TOOL)

N126 U-0.1 W-0.05 F0.002

N127 X-0.03 F0.004

N128 G00 X2.65

N129 X5.5 Z2.0 T0800 M09

N130 M30

%

In block N122, the tool is positioned 0.015 past the position Z-2.0. Block N123 makes only a temporary groove - to 2.2. The next block (N124) is a motion out of the temporary groove, to the starting diameter of the chamfer 2.46. In the following block N125, tool shifts in the Z-axis, to the chamfer start position. The amount of 1.950 was calculated by additions and subtractions:

$$1.875 - 0.020 - 0.030 + 0.125 = 1.950$$

The amount of 1.875 is the part back face (as per drawing), 0.020 amount is the chamfer size; 0.030 is the clearance, and 0.125 is the insert width. Note the 0.125 tool width position adjustment, to maintain the tool reference point on the *left* side of the cutting edge while actually cutting with the right edge. Block N126 is the chamfer cutting, using incremental mode. Using incremental mode saves a few calculations. If using absolute mode, block N126 will be calculated:

N126 X2.36 Z-2.0 F0.002

Also note the decreased feed rate for the chamfer only, to assure a good finish. Feed rate decrease can be quite significant for very small chamfers. The program remaining blocks are unchanged.

In some cases, two tools can be justified for part-off operations. Setup of the two tools has to be accurate. A

small, rigid grooving tool can do the startup groove and the chamfer, then a part-off tool can do the rest. At the completion of the part-off, bar stock projecting from the spindle will have a small step (tip). Make sure to program a facing cut for each subsequent part to take this step (tip) into consideration.

Preventing Damage to the Part

When the part is separated from the bar, it falls down. On impact, it may suffer a damage severe enough to make a good part a scrap. To prevent the possibility of a damage, CNC lathe operator may want to place a bucket filled with cutting coolant in the path of the falling part. Another method is to offset the part-off tool away from the centerline, just far enough that it does not separate the part. Then, when the machine is stationary (*i.e.*, not moving), CNC operator breaks the part manually.

In any case, always follow safety rules of the company they exist for your protection.

Never touch the part while the program is in operation, or the spindle is rotating

The best solution for part damage prevention is a CNC lathe equipped with a parts catcher, which is often a special machine option, ordered at the time of machine purchase.

For part-off, just like for grooving operations, always make sure there is an adequate supply of inserts on hand. Tools with sharp edges, or with very small radii, are generally weak tools, yet doing some very demanding work. Nobody wants to run out of tools in the middle of a very important rush job.

30. SINGLE POINT THREADING

Threading is a machining process used to produce a helical groove of a given shape and size, usually on a cylinder (straight thread), or on a cone (tapered thread). The major purpose of threads is to connect two parts together without damage during joining and disjoining (assembly and disassembly). The most common applications of threading fall into four major categories:

Fastening devices ... screws, bolts and nuts

Measuring tools ... micrometer barrel

Motion transmission ... lead screw, camera lenses

Torque increase ... lifting or supporting jacks

Thread cutting is a very versatile manufacturing process. There are two main groups of thread production - metal cutting and plastic molding. It should not be a surprise that it is the plastic molding method that dominates manufacturing industry. Given the number of detergent bottles, pop bottles and other plastic products we consume, the number of threaded products using this method is astronomical.

In metalworking area of thread production - the primary group of interest - there are smaller several subgroups:

Thread rolling

Thread forming

Thread grinding

Tapping and die work

Thread milling

Single point thread cutting

For a typical CNC programmer, the areas of interest are usually confined to the last three - tapping (but no die work), thread milling and single point threading. Methods for tapping operations have been described in *Chapters 25 and 26*, thread milling will be described in *Chapter 45*. This chapter covers programming methods described as *single point threading*.

A. THREADING ON CNC LATHES

In a *single setup*, CNC lathes can produce very high quality threads, in addition to turning, boring, grooving and other operations. This is a very attractive feature for manufacturers and many machine shops have purchased a CNC lathe for that reason alone. Any secondary operation requires additional setup, increasing the cost of production.

Single point thread cutting - typically known as a *single point threading* - uses a tool holder similar to other tool holders, but contains a special threading indexable insert, which may have one, two or three tips. Generally, the threading insert shape and size must match the shape and size of finished thread - *Figure 38-1*.

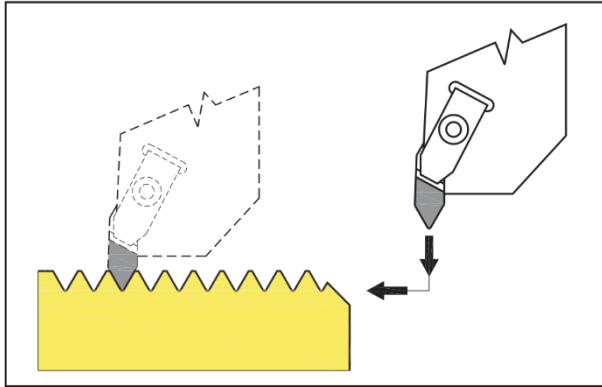


Figure 38-1

Comparison of the thread form and the threading tool shape

By definition, standard single point threading is a unique machining process of cutting a helical groove of a specific shape with uniform advancement per spindle revolution. The thread shape - or *form* - is mainly determined by the shape and mounting position of cutting tool. The uniformity of advancement is set by programmed feed rate.

Form of a Thread

The most common thread form used in CNC programming is the familiar V-thread (in the shape of letter V) with a 60° included angle. There is a large variety of V-shape thread forms used in manufacturing, including metric and imperial threads. Other forms include trapezoidal shapes, such as metric trapezoid, Acme and worm threads, square and round threads, buttress threads and many others. In addition to these relatively common forms, there are threads specific to a particular industry, such as automotive, aircraft, military, and petroleum industries. To make matters even more interesting, the thread shape can be oriented on a cylindrical surface, a conical surface, it also can be external or internal. The thread can be cut on a face (scroll threads), even on circular surfaces. It can have a single or multiple starts, right or left hand orientation, constant or variable lead, etc. *Many programmers joke that CNC threading can be a breeze or a nightmare with nothing between.*

Threading Operations

This section contains a detailed list of various threading operations that can be programmed for a common CNC lathe. Several operations require a special type of threading insert and some operations can only be programmed if the control system is equipped with special (optional) features:

Constant lead threads

Variable lead threads

External and internal threading

Cylindrical threads (straight threads)

Tapered threads (conical threads)

Right Hand (R/H) and Left Hand (L/H) threads

Face threads (scroll threads)

Single-start threads

Multi-start threads

Circular threads

Multi-block threads

In spite of the seemingly endless possibilities and combinations in thread cutting, the programming knowledge and experience gained in one category will be indispensable in other categories. A good threading program is based on a sound knowledge of common threading principles.

B. TERMINOLOGY OF THREADING

Threading is a relatively large subject, in fact, it is large enough to have a whole book dedicated to it. As subjects of this kind usually are, threading has its own technical terms. These terms appear in many books, articles, technical papers, manuals and other sources. To understand them is a requirement for any CNC programmer and operator.

Listed here are some of the most common terms used for threads and thread cutting:

ANGLE OF THREAD

... is the included angle between two thread sides, measured in an axial plane (along an axis)
(also see ROOT)

CREST

... is the top surface of a thread that joins the two sides

DEPTH OF THREAD

... generally, the distance between the crest and root of the thread, measured normal to the axis (in programming, depth is considered as a measurable size per thread side)

EXTERNAL THREAD

... is a thread that is cut on the outside of machined part, for example, as a screw or a bolt

INTERNAL THREAD

... is a thread that is cut on the inside of machined part, for example as a nut HELIX **ANGLE**

... is the angle made by helix of the thread at the pitch diameter with a plane perpendicular to the axis

LEAD

... is the distance the threading tool will advance along an axis during one spindle revolution. The lead always determines the threading feed rate and can have constant or variable form

MAJOR DIAMETER

... is the largest diameter of the thread

MINOR DIAMETER

... is the smallest diameter of the thread

MULTISTART THREAD

... is a thread with more than one start, shifted by the pitch amount

PITCH

... is the distance from a specified point of one thread to the corresponding point of the adjacent thread, when measured parallel to machine axis PITCH **DIAMETER**

... on a straight thread, the pitch diameter is an imaginary diameter where both external and internal diameters contact each other (*i.e.*, diameter where they meet) ROOT **(also see CREST)**

... is the bottom surface of a thread, joining the sides of two adjacent threads

SCROLL THREAD

... is also known as a face thread - it is a thread machined along the X axis, rather than the more

common thread machined along the Z axis ^{SHIFT}

... in multi-start threading, it is a distance by which the cutting tool is displaced to cut another start; this distance is always equal to the thread pitch. The number of shifts is always one less than the number of starts

TAPERED THREAD

... is a thread on which the pitch diameter is increased or decreased by a constant ratio (such as a pipe thread)

TPI

... in imperial units of measuring, the number of threads counted over the length of one inch (1 / pitch) - metric thread is defined by its pitch (TPI equivalent is not applicable)

C. THREADING PROCESS

Threading is one of the most automated programming tasks in modern machine shop, yet it could be one of the more difficult operations done on CNC lathes. At first look, it may seem an easy procedure to make a program for toolpath that has all cutting parameters clearly defined, such as threading. Practical applications, however, could present a big departure from theory. This comment may be arguable, at least until it is time to start searching for solutions to unusual threading problems or even regular threads that *'just don't seem to be coming out right'*. An experienced programmer should have the ability to think of yet another solution, when all other solutions seem to have been used up. Such approach is the key to any problem solving and applies equally to threading problems.

What often causes threading to be a difficult operation is the cutting tool application. *Single point threading tool is unlike any cutting tool*. Although threading tool holder is mounted in the turret just like other tools, thread cutting insert is unique. Threading tool does not only *cut*, but it also *forms* the thread shape. Typically, any threading insert will have the shape of finished thread. Mounting a threading tool in the turret can be done at 90° *to* - or *parallel with* - the machine spindle centerline, *regardless* of the thread being cut. Decision which way to mount the tool is determined by the thread angle, relative to the spindle center line. It is important that the tool is mounted *square* with the turret face. Even a small angular deviation will have an adverse effect on the finished thread.

Steps in Threading

When comparing a common V-type 60 threading insert with equally common 80° insert commonly used for rough turning/boring, some interesting results will emerge:

Description	Threading tool: 60 included angle	Turning tool: 80 included angle
Tool nose radius	Almost sharp tip	0.8 mm (0.0313")
Included tool angle	60 + WEAK support	80 + STRONG support
Typical cutting feed rate	Up to 6.5 mm/rev + (0.25 in/rev) +	0.4 - 0.8 mm/rev (0.015 - 0.03 in/rev)
Typical cutting depth	Very SMALL	MEDIUM to LARGE

Such comparisons are not fair, and the table is definitely not scientific, but it proves one important conclusion for threading - *the weakest tool does the heaviest machining*. Yes, welcome to the world of single point threading.

Indirectly, the table shows that even a very fine pitch thread *cannot* be machined using a single threading pass only. A single pass would produce a thread of very poor quality at best and - most

likely - a totally unusable thread. Tool life would also suffer greatly. Much better approach is to program the thread as a series of small cutting passes, where each pass increases the thread depth.

Spindle Synchronization

Unlike any other CNC lathe machining operation, single point threading requires that the thread cutting tool starts machining at *exactly* the same circumferential point for each depth of cut. This requirement is called *spindle synchronization*. In programming, this is not an issue at all, neither it is an issue during setup and normal machining. When this knowledge is very important can be summed up in two points:

Threading process - general understanding

Thread recutting - rework

As for the first item, understanding any subject in depth is always rewarding, thread cutting included. All new knowledge leads towards change of thinking, hopefully for the better. This is true in both programming and machining - in threading, solid knowledge can be used to solve various problems that may occur.

It is the second item that every CNC programmer - and *particularly the CNC operator* - should fully understand recutting a thread. The subject of *thread recutting* is covered at the end of this chapter. At this point, just keep in mind that once a threaded part is taken out of the holding device, it will be extremely difficult - if need be - to put it back for recutting into the same exact position as before.

With practice and a really 'good eye', it can be done. The best way to avoid recutting is prevention - measure the thread while it is still mounted, there are several methods.

Individual Threading Motions

Previous paragraphs have established that for multi-pass single point thread cutting, machine spindle rotation must be synchronized for each threading pass, so each thread start point is located at the same position on the threaded cylinder. High quality thread will be completed when the last cutting pass produces proper thread size, shape, surface finish and tolerances. Since single point threading is a result of several passes cutting a single thread, programmers must understand each of these passes well - they constitute *individual threading motions*.

In programming, basic structure of each pass remains the same, only the actual thread data change from one pass to another. In a most elementary setup, there are at least four motions for each threading pass - the table example shows one-pass threading as applied to a standard straight thread:

Threading Motion	Motion description
1	From the starting position, move tool to the thread diameter in rapid motion mode
2	Cut the thread - one axis thread cut (at the feed rate equal to the lead)
3	Rapid retract from the thread
4	Rapid return to the starting position

Expanding on these brief descriptions, the four step tool motion process will typically include at least four following considerations that are critical to the CNC program.

Start Point for Threading

Before the actual first motion, the threading tool must move from its current position (for example, from indexing position) to a position close to the machined part. This is a rapid motion,

in the air (G00 motion). Make sure to calculate the XZ coordinates for this position correctly. The coordinates are called the thread *start position*, because they define where the thread cut will start from and eventually return to. Such a start position must be defined *away* from the part, but still close to the thread, as an intersection of the X axis clearance and the Z axis clearance.

For the X-axis, selection of a clearance greater than the thread lead is usually recommended. This is a *physical* clearance, per side - it has to be doubled to define the actual coordinate. For the Z-axis (horizontal thread is assumed), there has to be a space before the tool contact part material. This clearance necessary is for acceleration and the general rule is for about three times the thread lead. More details follow in the next section.

Threading Motion 1 - Rapid mode

The first tool motion is directly related to the thread itself. It is a motion *from* the start position *to* the first thread cutting diameter. Since the thread cannot be cut at full depth in a single pass, the total depth must be divided into a series of more manageable depths. Each depth will depend on the type of tool, material used and overall setup rigidity. This approach motion is a rapid mode.

Threading Motion 2 - Threading mode

When the threading tool reaches selected cutting diameter for a given depth, the *second motion* becomes effective. This is the actual threading pass - it will be cut during this step, at a specified feed rate and only when machine spindle is *synchronized* with the threading feed rate. There is no need to take any special steps to maintain the synchronization - in any threading mode, it is automatic. Thread will be cut to the programmed thread end position, usually along the Z-axis.

Threading Motion 3 - Rapid mode

The *third threading motion* follows the actual thread cut *after* the programmed diameter is fully completed. Then, the threading tool must retract from the thread - at machine rapid rate and to a specified X-axis clearance position. Normally, this should be the same position as X-axis *start position*. It is the most logical position and always provides expected and verified results. X-axis clearance is always a diameter programmed *outside* of the thread area.

Threading Motion 4 - Rapid mode

With the final *fourth motion*, one-pass threading process is completed. There is a very important difference from the X-axis clearance in this motion - as expected, in the fourth motion, the threading tool *must* return along the Z-axis but where? *Always* to the Z-axis start position. *There is no choice*. Returning to any other point will result in a scrap, and, possibly, a broken tool. The reason? While the initial Z-position in *Motion 1* was selected with only acceleration in mind, it automatically became the control position for spindle synchronization. Therefore, the return to the same position is absolutely mandatory. Once correctly established, the same Z-axis start/end position *guarantees* the only way to maintain spindle synchronization that is necessary for threading. In fact, careful manipulation of this position is very useful in multi-start threading (described later in this chapter). *Motion 4* is always in rapid mode.

Remaining Passes

All remaining passes are calculated using the same approach, using any new thread diameter for exact thread depth control. Note that only *Threading Motion 2* will be programmed in threading mode, using a proper G code. Threading motions 1, 3 and 4 will be in rapid mode.

This typical description (illustrated in *Figure 38-2*) is only general in nature and usually not sufficient by itself for high quality thread cutting.

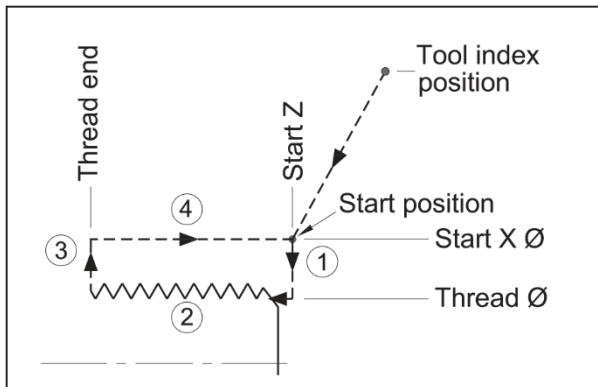


Figure 38-2

Basic steps in single point thread cutting

Thread Start Position

In reality, the tool start position is a specially selected *clearance position*. Unlike for turning or boring, this position has a special significance in threading. For a straight cylindrical thread, the minimum suitable clearance along the X-axis is about 2.5 mm (0.1 inches) per side, more for coarse threads, usually not smaller than the thread lead. For tapered threads, clearance calculation is the same, but applied over the *larger* diameter (external threads).

As for the clearance along Z-axis, some special considerations are necessary. When the threading tool comes into contact with material, it must be advancing at exactly 100% of the programmed feed rate. Since the cutting feed rate for threads is *always* equivalent to the thread *lead*, which may be quite large, certain amount of time will be required to reach the 100% of programmed feed rate. Just like a car needs some time to accelerate before reaching its cruising speed, threading tool has to reach full feed rate *before* it contacts any material. Real effect of acceleration must always be considered when selecting this clearance amount.

When programming coarse threads, the front clearance amount required will generally be much greater than a similar amount for fine or medium threads. For example, a common thread with 3mm pitch (8 TPI) will require feed rate of 3 mm/rev (0.125 in/rev). If the Z-axis clearance is too small, machine acceleration process will be incomplete upon the tool contact with material. Final result will be an imperfect and unusable thread. To avoid this very serious problem, a simple rule may help:

Z-axis start point clearance should be three to four times the length of thread lead

This is only a rule of thumb, a general rule; it works well in everyday practice. Control manuals may offer a scientific way of calculating the minimum clearance, but this extra effort is seldom necessary.

Obstacles for Clearance

In some cases, the Z-axis clearance must be reduced because of space shortage, such as when thread cutting starts very close to a tailstock or machine limits. As acceleration time depends directly on the spindle speed (r/min), one common solution to avoid imperfect threads in this case is to *reduce* the spindle speed (r/min) - *feed rate must **not** be reduced* - it represents the thread lead. For more complex methods of infeed, Z-axis start position will be changed for each cut by a calculated amount.

Another method to increase Z-axis clearance is to modify the actual threading holder and/or insert, by grinding-off its less important parts. It may be a crude way, but it works in emergency.

Thread Cutting Diameter and Depth

For cylindrical and conical thread cutting using the block method of programming (no cycles), each thread cutting diameter must be selected. From thread start position, the cutting tool will

move *towards* spindle centerline for external threads and *away from* spindle centerline for internal threads. Actual cutting diameter for each pass must be selected not only with respect to the thread diameter, but also with respect to machining conditions.

In threading passes, the insert chip load becomes heavier as cutting depth increases. A damage to the thread, to the insert, or both, can be averted by maintaining a *consistent chip load* on the insert. One way to achieve consistency is to *decrease* each subsequent depth of the thread, another way is to apply a suitable *infeed method*. Commonly, both of these popular threading techniques are used simultaneously, for even better results.

To calculate depth of each threading pass, complex formulas are not required, just some common sense and a bit of experience. All threading cycles have an algorithm (special process) built in the control system that calculates each depth automatically. For manual calculations, the procedure follows a logical approach. *Total thread depth* (measured per side) must always be known - part programmer decides how many *threading passes* will be suitable for a particular thread.

Another threading amount to be decided is the *last cut depth*, a cut that actually finishes the thread to size. These amounts are usually very small and come from experience. In addition to the last cut, some threads may require a repetition of the last cut at the same depth - this cut is called the *spring pass*. Its main purpose is to remove any material left due to tool deflection caused by pressures. The rest is limited to mathematical calculations or available charts - and the control system.

When all three parameters (programmed amounts) have been firmly established, the total cutting depth must be distributed (divided) among individual threading passes, including the last pass depth. That will require a decision how many threading passes are necessary?

Number of Threading Passes

If each threading pass is programmed individually, rather than using threading cycles, programmer must decide how many passes are required for a given thread. This is a very important decision - too few or too many passes may cause severe problems at the machine. There is no easy way to change such a program - in almost all cases, an erroneous program will have to be re-written from scratch (at least its threading portion). The number of passes required is often listed in tooling catalogues and provides a good start. Otherwise, consider the tool material, material being cut, lubrication used, method of setup, length of part, rigidity, etc.

Thread Depth Calculation

To fully understand how depth calculation works in CNC threading, two types of depth should be identified:

Total thread depth

Depth of each threading pass

Total thread depth is always measured 'per side' (radius value) and can be calculated from the *major* and *minor* diameters, if they are known:

Major	Minor
Thread Depth	_____
2	

More than likely, only the *major* diameter will be known, as per drawing definition. In this case, a common formula can be used to calculate thread depth - just keep in mind that there are actually *two similar* formulas - one for *external* and one for *internal* threading.

For the most common 60 V-thread in ISO M (metric) and UN (Unified National) forms - and - *external* threads only, the formula is:

0 61343.
$D_{EXT} \quad \text{———} \quad 0 \ 61343. \quad P$
TPI

For the same group of threads applied *internally*, the formula is similar:

054127.
$D_{INT} \quad \text{———} \quad 054127. \quad P$
TPI

+ where ...

D_{EXT} = Single depth of external
ead

D_{INT} = Single depth of internal
ead

TPI = Number of threads per inch

P = Thread pitch in mm *or*
1/TPI for imperial

An example of actual usage is included in this chapter.

Depth of each threading pass should always be calculated for best results. In emergency, an approximation of individual cuts - if done properly - will also work well. The main goal here is to select the first cut depth, then *decrease* each depth after (for subsequent passes). Consider the reasoning here - the threading tool is triangular. When the first depth is cut, a certain area of the triangle will be cutting. As the insert is engaged deeper, more of its area will be included, potentially creating a situation when *too much* material is removed. Programming each 'next' cut with progressively decreasing depth is the main key to successful threading. These cuts take place between *major* and *minor* diameters, never deeper.

Thread Depth Constants

The two common constants for calculating thread depth are **0.61343** for external threads, and **0.54147** for internal threads. Both are applied for any 60° V-thread ISO metric threads or UN imperial threads. Although it is not important to know the source of these constants, understanding their origin helps to better understand this most common thread form. It should be expected that these constants have been calculated from actual thread data, based on established international standards - *Figures 38-3 and 38-4*.

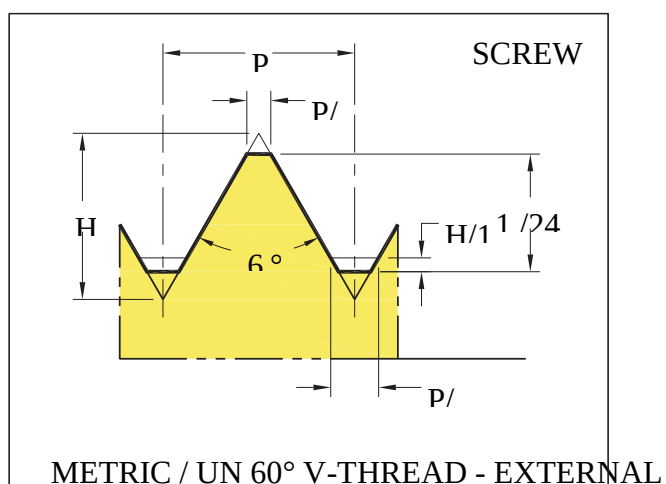


Figure 38-3

Standard ISO (M) and imperial (UN) 60 V-thread form - **EXTERNAL**

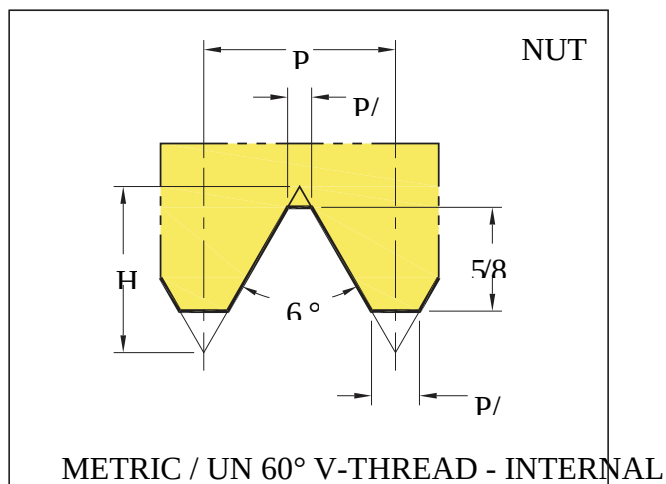


Figure 38-4

Standard ISO (M) and imperial (UN) 60 V-thread form - **INTERNAL**

As with many constants in mathematics, these two are also based on the main unit being 1 (one). In this case, since the thread depth will change with any pitch change, the pitch is assumed to have a value of 1. That means $P=1$ mm or $P=1$ inch. There is a reason why the unit of one has been used - it allows multiplication by *any* actual unit value to produce the required result. The formula using such constant is usable for any pitch size. One other dimension is also important - the dimension H , overall thread height, measured between sharp points.

If pitch $P = 1$,

$$\text{then } \dots H = 0.5 / \tan 30^\circ = 0.866025$$

From here the actual calculations of thread depth can be made, based on the fractional dimensions of the standard:

$$\begin{aligned} 17/24H &= 17/24 \cdot 0.866025 \\ &= 0.708333 \cdot 0.866025 \\ &= \underline{0.613435} \dots \text{EXTERNAL} \end{aligned}$$

$$\begin{aligned} 5/8H &= 5/8 \cdot 0.866025 \\ &= 0.625 \cdot 0.866025 \\ &= \underline{0.541266} \dots \text{INTERNAL} \end{aligned}$$

American National Thread

Another standard - an older one - may still be used in various publications and drawing specifications, so it is important to understand this thread form and be able to calculate its thread depth. *American National* standard does not use the constant of 0.613 for external thread depth, but constant of 0.649 - see Figure 38-5.

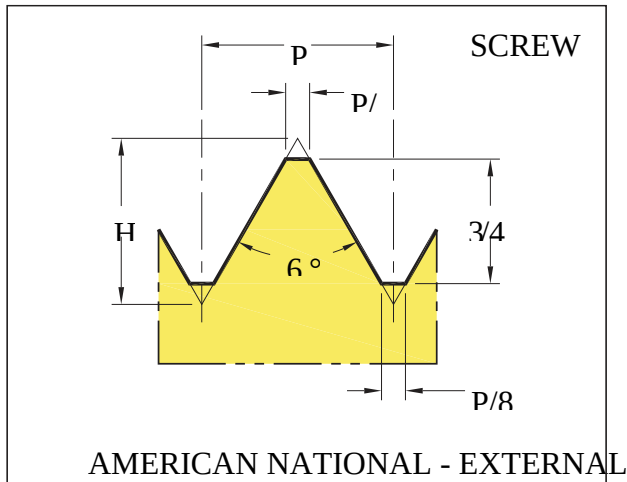


Figure 38

American National thread form

The main difference between the two forms is the specified height of $\frac{3}{4}H$ compared to $\frac{17}{24}H$. The overall depth H is still the same - 0.866025.

$$\frac{3}{4}H = \frac{3}{4} \cdot 0.866025$$

$$= 0.75 \cdot 0.866025$$

$$= \underline{0.649519} \dots \text{EXTERNAL}$$

Constant for internal threads remains the same, at 0.541.

Calculation of Threading Passes

When individual threading passes are only approximated (meaning they are 'estimated'), they may - or they may not come out right. When the same passes are actually *calculated*, it does not automatically mean the thread will come out right either, but the chances are much better. There are many technological issues to deal with that cannot be included in any mathematical formula.

To illustrate calculations of individual passes, a typical example will be used - a simple external thread illustrated in Figure 38-6.

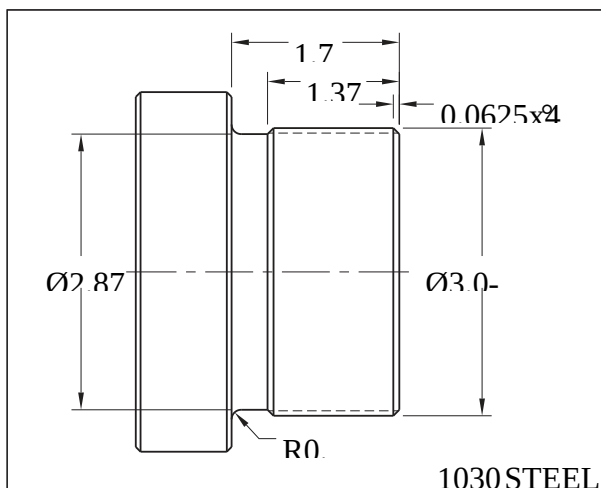


Figure 38-6

Sample part used for thread depth calculation

- only important dimensions are shown

Calculating each threading pass is only necessary for the so called 'long hand' threading (where each motion has its own block), they are *not* needed for threading cycle G76.

As always in problem solving, the first step is to isolate *known data* relevant to the solution. In this case, they are:

Major (given) diameter ... 3.0 inches

Number of threads per inch ... 12 TPI

Length of thread ... 1.5 (as reasonable)

Material ... Mild steel

From the data that is known, some additional data can be calculated - *or* - selected:

Thread depth ... $0.61343 / 12 = 0.0511$

Minor (final) diameter ... $3 - 2 \cdot 0.0511 = 2.8978$

First cutting depth ... 0.021 (selected)

Cutting feed rate ... Thread lead = 0.0833

All these settings provide enough information to continue with program development.

Well, there is one more programming decision to make yes, it is optional, but also strongly recommended - is to select a *last pass depth* - known as *finishing allowance* - for example, 0.004 for the following calculations. Now, all data are in, so the number of passes can be calculated:

$$n = \frac{P - R}{Q}$$

+ where ...

n = Number of threading passes

P = Depth of thread per side

R = Finishing allowance amount (last pass depth)

Q = Depth of the first threading pass

It appears that the selected letters do not suggest any specific meaning. That is true, but at this point, consider them to be just letters identifying known or unknown thread related values. Yet, they were selected carefully, with threading cycles in mind, covered later in this chapter. For the example, the calculated number of passes will be:

$$n = \frac{0.0511 - 0.004}{0.021} = 5.035 \text{ passes}$$

Accepting that *five threaded passes* are suitable for this job (excluding the finishing allowance), they will be distributed between the specified diameter of 3 inches and the calculated final diameter of 2.8978 inches, with the first depth pass of 0.021. The depth of cut used for *calculation* has to be reduced by the finishing allowance, which will become the sixth pass. So, the usable depth is not 0.0511 but $0.0511 - 0.004 = 0.0471$. The 0.004 will be an *additional* pass, which is optional but has been selected here.

Pass 1 = 0.021 1 = 0.0210

..cutting depth = 0.0210

Pass 2 = 0.021 2 = 0.0297

..cutting depth = 0.0297

Pass 3 = 0.021 3 = 0.0364

..cutting depth = 0.0364

Pass 4 = 0.021 4 = 0.0420

..cutting depth = 0.0420

Pass 5 = 0.021 5 = 0.0470

..cutting depth = 0.0470

Pass 6 = Pass 5 + 0.004

..cutting depth = 0.0510

Cutting depth shows *accumulated* depth. In order to calculate individual threading diameters, subtract each depth twice (double depth) from the major diameter of 3 inches:

Cutting diameter 1	..	3.0 - 2 0.0210	=	2.9580
Cutting diameter 2	..	3.0 - 2 0.0297	=	2.9406
Cutting diameter 3	..	3.0 - 2 0.0364	=	2.9272
Cutting diameter 4	..	3.0 - 2 0.0420	=	2.9160
Cutting diameter 5	..	3.0 - 2 0.0470	=	2.9060
Cutting diameter 6	..	2.906 - 2 0.004	=	2.8980

Note that this particular example shows almost perfect calculations (within 0.0002) - that is not always possible, and some adjustments usually be necessary. Arbitrary selections or even manual adjustments of one or more depths may have to be made for best results. *Figure 38-7* shows in scale - the distribution of threading cuts for the example.

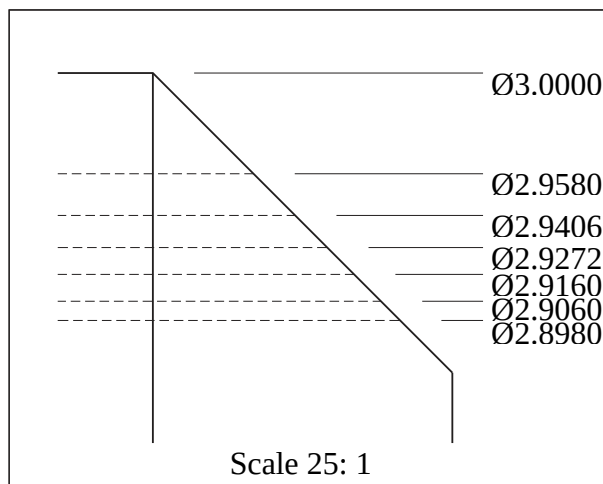


Figure 38-7

Distribution of individual threading passes based on a formula

All this theory has to make some practical sense - it is time to apply the theoretical principles into actual program.

Thread Cutting Motion

When a threading tool reaches selected threading pass depth, it is ready to cut the thread itself. All thread cutting motions originate from the start position (start point). In fact, this start point should also be the *end point* for each threading pass, so it can become the consistent start point for the next pass. During thread cutting, a feed rate must be active. Although the threading cut is, in effect, a linear motion, do not *ever* use preparatory command G01 for threading. If G01 is used, the start point for each pass will *not* be synchronized with the previous thread start. Instead of G01 command, use a G code specifically designated for threading. Preparatory command G32 is the most common code used by Fanuc controls for threading without cycles. During a thread cutting motion with G32, control system automatically disables the feed rate override. CNC operator has to be extra careful to set the threading tool exactly, particularly when thread ends close to shoulders of the part or close to the chuck. To illustrate the programming process up to this point, here is a typical program section:

...

N61 G00 X3.3 Z0.25 (START POINT XZ)

N62 X2.958 (THREAD DIA NO. 1 START) N63 G32 Z-1.6 F0.0833 (THREAD TO END)

...

When block N63 is completed, threading tool is positioned at the end of the thread at X2.958 Z-1.6.

Retract from Thread

Once the tool has reached its end position along Z axis, the tool must leave the material immediately, to avoid making any physical damage. This program block represents the *third* motion within the basic threading process. Retract motion can have two forms - straight out in one axis (normally along the X-axis), or a gradual pull-out along two axes (simultaneously along XZ axes) - *Figure 38-8*.

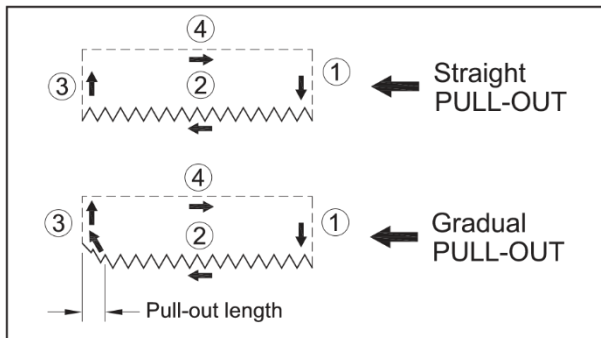


Figure 38-8

Exit options from a completed thread

Generally, the *straight pull-out* should be programmed whenever thread ends cutting in an *open space*, for example, in a relieve groove or a recess. For threads that do not end in an open area (some shaft applications, for example), the *gradual pullout* is a better choice. Gradual pull-out motion produces better quality threads and prolongs threading insert working life. In order to program a straight pull-out, the threading mode G32 must be canceled and replaced by a rapid motion mode, using the G00 command:

N64 G00 X3.3 (RAPID OUT)

For any gradual pullout, the threading G code and the feed rate *must remain in effect*. When the normal length of thread is completed - but *before* the tool is retracted - the threading tool moves in two axes simultaneously, ending *outside* of the thread. The normal length of the pullout is usually 1 to 1-1/2 times the lead (not the pitch), the suggested angle is 45°. It is also important to pay attention to the clearance diameter.

N64 U0.2 W-0.1 (GRADUAL PULLOUT-THREADING ON)

N65 G00 X3.3 (RAPID OUT)

For *external* threads, the rapid-out clearance diameter must always be *further* away from spindle center line than the last diameter of gradual pullout. For *internal* threads, the clearance diameter must be *closer* to spindle center line than the diameter of gradual pull-out.

In G32 mode, the gradual pull-out can be within the program length or can extend beyond it. The programmer has a choice here, it is based on the job requirements.

Return to Start Position

Regardless of how the threading tool retraction is programmed - straight or gradual - the last step is always a return to the starting position. This tool motion is entirely in the open space, therefore programmed in the rapid motion mode G00. Normally, this return motion to the starting point is along one axis only, usually the Z axis. This is because in most programs, the tool retraction from the thread has already reached the X-axis diameter and only the Z-motion is left.

Here is a program segment showing gradual pull-out:

...

N61 G00 X3.3 Z0.25 (START POINT XZ)

N62 X2.958 (THREAD DIA NO. 1 START)

N63 G32 Z-1.6 F0.0833 (THREAD TO END)

N64 U0.2 W-0.1 (GRADUAL PULL-OUT)

N65 G00 X3.3 (RAPID OUT) N66 Z0.25 (RETURN TO Z-START)

...

Returning to the start point makes sense - this point provides consistency in thread programming.

D. THREADING FEED AND SPINDLE SPEED

Regardless of any active *cutting* motion, there is always a feed rate involved - this feed rate controls how effectively the tool removes material in various operations - drilling, contouring, facing, pocketing and many other machining operations. As threading operations are the main subject of this chapter, it is important to understand how feed rate is related to single point threading.

In threading, the choice of cutting insert selection, spindle speed and threading feed rate are always restricted more so than for any other machining operation. Both, the threading tool and feed rate are determined by the engineering requirements - as drawing specifications. In terms of material removal, threading insert is one of the *weakest* tools used on CNC lathes - yet its applications demand some of the *heaviest* feed rates used in CNC lathe programming for any tool.

Other factors that can influence the final thread have to be dealt with as well, such as spindle speed (both minimum *and* maximum), the depth of each threading pass, the tool edge preparation, thread infeed method selected, setup of cutting tool and insert, plus many similar considerations. Often, a simple change of only one factor will correct a threading problem. At other times, many factors have to be considered - and possibly changed - to produce the best thread possible.

Figure 38-9 compares feed rates for turning and threading - they are both in feed rate per revolution.

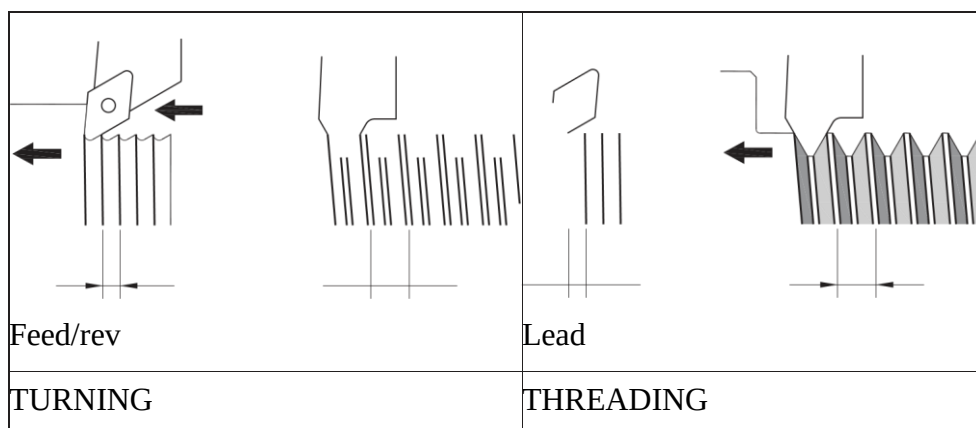


Figure 38-9

Comparison of feed rates between turning and threading

Next, we look at some practical consideration when selecting feed rates for threading.

Threading Feed rate Selection

Selection of feed rate for general turning, boring, grooving, etc., is based on such factors as

material type, tool nose radius, required surface finish, etc. In this sense, the ‘correct’ feed rate for such operations cover a large range. In threading, this flexibility is limited. Threading feed rate is *always* determined by the *lead* of the thread - *never* the pitch. In imperial units drawings, thread description is provided as the number of threads over one inch length, or TPI (TPI = threads per inch), and a nominal diameter. As an example, a thread that is described in the drawing as 3.75-8, means that the thread has *8 threads per inch*, and the nominal diameter (for example, the major diameter) is 3.750. All single start *metric* threads have the pitch standardized, depending on the thread diameter. For instance, a thread described as M243 is a single start metric thread with the pitch of 3 mm on a 24 mm nominal diameter. A description M70.75 is a single start thread with three quarters of a millimeter pitch and nominal diameter of 7 mm.

Regardless of dimensional unit used, the most important terms for selecting the correct feed rate are the *lead of thread* and the *number of threading starts*.

It may help to review basic relationship between *thread lead* and *thread pitch* (see details in terminology of threading on *page 350*). In a common machine shop conversation (shop talk), the words *lead*, and *pitch* are often used incorrectly. The reason is that for a single start thread, the amount of lead is *identical* to the amount of pitch. Since most machine shops work with a single start thread on a daily basis, misuse of the terms is seldom noticed. In addition, virtually all taps have a single start. What may be acceptable in a shop talk language has to be interpreted correctly in CNC programming. Each term has a very specific meaning in threading, so use them in the correct way:

	No of starts.	
LEAD	_____	Feed rate
	TPI	

The programmed threading feed rate is always the lead of thread, never the pitch!

	— 1	
PITCH	_____	
	TPI	

From these two formulas it is easy to deduct that if the number of starts is *one*, both the lead and pitch will always have the same value.

The following formula should be applied for threading feed rate calculation:

F	L	P	n

+ where ...

F = Required feed rate (mm/rev or in/rev) L = Lead of the thread (mm or inch) P = Pitch of the thread (mm or inch) n = Number of starts (positive integer)

For example, a thread with a single start and the pitch of three millimeters (3 mm) will require feed rate of

3 1 = F3.0

For threading programs that use imperial units, the above formula is equally valid, since

P — 1	
TPI	

+ where ...

P = Thread pitch

TPI = Number of threads per inch

As an example, the thread with one start and 8 TPI will require feed rate of

$$1/8 \text{ in} = 0.125 \text{ in} = F0.125$$

Multi-start threads are special in many ways, but the feed rate selection is also the *lead* - not the pitch of the thread. For example, a double start thread with 12 TPI will have the following feed rate:

$$1/12 \text{ in} \times 2 = 0.083333 \text{ in} = F0.166667$$

For most Fanuc controls, imperial threading feed rate can be programmed with six digits after the decimal place.

Spindle Speed Selection

Spindle speed for thread cutting is always programmed in direct *r/min*, never as constant surface speed (CSS) (cutting speed). That means G97 preparatory command must be used with address S, specifying the number of revolutions per minute. For example, G97 S500 M03, will result in 500 r/min spindle speed. As it is true that single point threading takes place over several diameters between major and minor, G96 selection would seem logical. This is not the case. First, even for fairly deep coarse threads, the difference between the first and last diameter is insignificant. Second - *and this reason is even more important* thread cutting requires a perfect spindle and feed rate *synchronization* at the start of each pass. Such synchronization can be more accurately achieved only with constant *r/min* rather than constant surface speed (CSS).

For majority of threads, the selection of *r/min* requires only consideration of general machining conditions, similar to turning operations. At the same time, selecting spindle speed should be considered along with the feed rate. Because of some heavy feed rates used for threading, there is a distinct possibility that certain threads cannot be cut at *any available spindle speed*. If this is confusing, keep in mind that feed rate is determined not only by the lead, but also by the *machine overall capability*. Every CNC lathe has a programmable feed rate value, specified in either *mm/min* or *in/min*, up to a certain maximum for each axis.

Take an example of a CNC lathe that has maximum programmable feed rate for X axis 7000 mm/min (275 in/min) and for Z axis 12000 mm/min (472 in/min). Consider the fact that there is a direct relationship between spindle speed and feed rate per revolution. Result of this relationship is actually *feed rate expressed in terms of time*, not per revolution. Feed rate per time is always result of the spindle speed in direct *r/min* multiplied by the feed rate per revolution in *mm/rev* or *in/rev*.

Metric units example:

$$700 \text{ r/min} \times 3 \text{ mm/rev} = 2100 \text{ mm/min}$$

Imperial units example:

$$700 \text{ r/min} \times 0.125 \text{ in/rev} = 87.5 \text{ in/min}$$

In CNC lathe programming generally, not only in threading, always make sure that feed rate per revolution combined with spindle speed will be *less than or equal to* the maximum *available* feed rate per time for the axis with the lower rating, which is usually the X axis.

Based on this simple rule, maximum spindle speed (in revolutions per minute) for a given lead can be selected according to the following formula:

	Ft
R _{max}	_____max
	L

+ where ...

R_{max} = Maximum allowable spindle speed (r/min)

$F_{t_{max}}$ = Maximum feed rate per time (X axis)

L = Thread lead

Metric units example:

If thread lead L is 2.5 mm and the maximum feed rate for X axis $F_{t_{max}}$ is 7000 mm/min, then the maximum programmable threading spindle speed R_{max} will be:

$$R_{max} = 7000 / 2.5 = 2800 \text{ rpm}$$

Imperial units example:

If thread lead L is 0.125 inches and the maximum feed rate for X axis $F_{t_{max}}$ is 275 in/min, then the maximum programmable threading spindle speed R_{max} will be:

$$R_{max} = 275 / 0.125 = 2200 \text{ r/min}$$

The maximum allowable spindle speed (r/min) only reflects capabilities of the CNC machine. Feed rate actually used in a program must also take into account various machining and setup conditions, just like any other cutting operation. In practice, the majority of actual programmed spindle speed (r/min) will be well below the maximum capacity of CNC machine tool.

Maximum Threading Feed rate

Selection of cutting feed rate in general was discussed earlier, starting on page 91. After studying the section on maximum r/min selection (spindle speed), it should not be surprising that similar limitations apply to the determination of *maximum threading feed rate* for a given spindle speed (programmed as r/min). Again, the limits of CNC machine tool are very important, so be aware of them when writing thread cutting programs.

Maximum programmable threading feed rate for a given spindle speed (in r/min) can be calculated from the following formula:

	F_t	
$F_{r_{max}}$	————— _{max}	
	S	

+ where ...

$F_{r_{max}}$ = Maximum feed rate per revolution for a given spindle speed S

$F_{t_{max}}$ = Maximum feed rate per time (X axis)

S = Programmed spindle speed (r/min)

Metric units example:

In a metric example, the maximum machine feed rate for X-axis is 7000 mm/min and the programmed spindle speed S is selected as 1600 rpm. In this case, the maximum programmable threading feed rate will be (in mm/rev):

$$7000 / 1600 = 4.375 \text{ mm/rev}$$

Result shows the maximum lead that can be safely machined at 1600 r/min must be less than 4.375 mm.

Imperial units example:

If the maximum machine feed rate along X-axis is 275 in/min and spindle speed S is selected as 2000 r/min, then the maximum programmable feed rate will be (in *in/rev*):

$$275 / 2000 = 0.1375 \text{ in/rev}$$

Therefore, the maximum thread lead that can be cut at 2000 r/min is 0.1375 inches, which allows about 8 threads per inch or finer.

Changing the spindle speed (feed rate remains the same) will allow programming coarser threads on the same CNC lathe. For example, if only 1500 r/min is selected instead of the 2000, the maximum lead will increase to 0.1833 inches or about 6 threads per inch.

All calculated values only indicate the control and machine actual capabilities and do not guarantee a safe job setup or even suitable machining speed

Lead Error

Normally, threading feed rate requires the address F, with up to three decimal place accuracy for metric threads (F3.3 format), and four decimal place accuracy for threads in imperial units (F2.4 format). The majority of threads are short, and this accuracy is quite sufficient. There is never problem for metric threads, regardless of thread length, because the thread is defined by its lead already in the drawing. For threads programmed in imperial units, the thread lead must be calculated from drawing provided threads per inch (TPI). For many imperial threads, lead is accurately calculated within four decimals available for the F address. A 10 TPI requires programmed feed rate of F0.1, 16 TPI requires programmed feed rate of F0.0625, etc. These are threads that divide TPI into one, within the four decimal places accurately, such as 8, 10, 16, 20, 40 to name the most common number of threads. Not all threads fall into this rather convenient group. For many other threads, the calculated value must be properly rounded off.

Take a 14 TPI thread, for example. Ideal threading feed rate should be $1/14=0.071428571$ inches per revolution. Rounded value used in the program should be F0.0714. Over a short thread length there is no noticeable error, and the thread is well within all tolerances. That is not true if the thread is unusually long, or the rounded value has been improperly calculated. An *accumulative error*, known as the *thread lead error*, may result in possible scrap, due to incorrect thread. By using rounded value of 0.0714, the loss is 0.000028571 inches for *each* thread revolution. Lead error over one inch (or more) can be easily calculated:

L_e	$(F_i$	$F_p)$	TPI

+ where ...

L_e = Maximum lead error per inch

F_i = Ideal feed rate (nine or more decimal places)

F_p = Programmed rounded feed rate

TPI = Number of threads per inch

Over one inch, the error in the example will be 0.0004 of an inch, over fifty inches it will be full 0.0200 of an inch.

Thread length in ches	11.5 TPI		14 TPI	
	F0.0870	F0.086957	F0.0714	F0.071429
1.0	0.000500	0.000006	0.000400	0.000006

2.0	0.001000	0.000011	0.000800	0.000012
3.0	0.001500	0.000017	0.001200	0.000018
4.0	0.002000	0.000022	0.001600	0.000024
5.0	0.002500	0.000028	0.002000	0.000030
10.0	0.005000	0.000055	0.004000	0.000060
25.0	0.012500	0.000138	0.010000	0.000150
50.0	0.025000	0.000275	0.020000	0.000300

The table above shows accumulative error for 11.5 TPI and 14 TPI and compares two feed rates for each, as used in a program for threading.

Rounding Error

Another example, somewhat more critical, is an incorrect rounding value. Ideally, a thread with 11.5 threads per inch should be programmed with the feed rate of 0.086956522. If this value is rounded to F0.0870, the accumulated error is

0.0005 per one inch and the error over 50 inches will be 0.0250 inches (see table). Even if the CNC machine does not allow six decimal places for threading feed rate, the correct rounding of the calculation is very important.

Compare the following rounded values and the errors they cause (11.5 TPI over 50 inches):

0.0869 ... error of 0.0325 0.0870 ... error of 0.0250 0.0871 ... error of 0.0825

What a difference for only 1/1000th of an inch rounding.

E-address

As you see from the previous table, Fanuc engineers recognized this potential problem early and introduced the address E for threading feed rate on their earlier CNC controls (Fanuc 6, for example). The benefit of using E address for threading is that it allows programming with six decimal places instead of the standard *four* for imperial threads (increased accuracy allowed for metric threads using the E address is seldom used). With proper rounding, the accumulative error is virtually negligible.

Using the same illustration of 14 TPI over 50 inches, the error for the whole length will only be 0.0003 of an inch, if the F0.0714 is replaced by E0.071429. The second example, using a thread with 11.5 threads per inch, should be programmed with the feed rate of E0.086957. The accumulated error over fifty inches will be only 0.000275 inches, a negligible error. Note that the table shows *F* not *E* address:

The latest CNC systems allow use of six digit accuracy for the F-address as well

The lead error or rounding error are always potential problems when programming long thread leads. Depending on the kind of threading applications in machine shop, accumulative error of the thread lead may be critical, or it may never be an issue to deal with.

E. TOOL REFERENCE POINT

Good tool setup is critical to good machining environment. While a good setup is important to all tools, it is even more important to maintain good setup of threading tools, both external and internal. Tool cutting edge has to be correctly oriented, securely mounted in the insert pocket and it has to be the right type. Its reference point, used for setup, is also very critical.

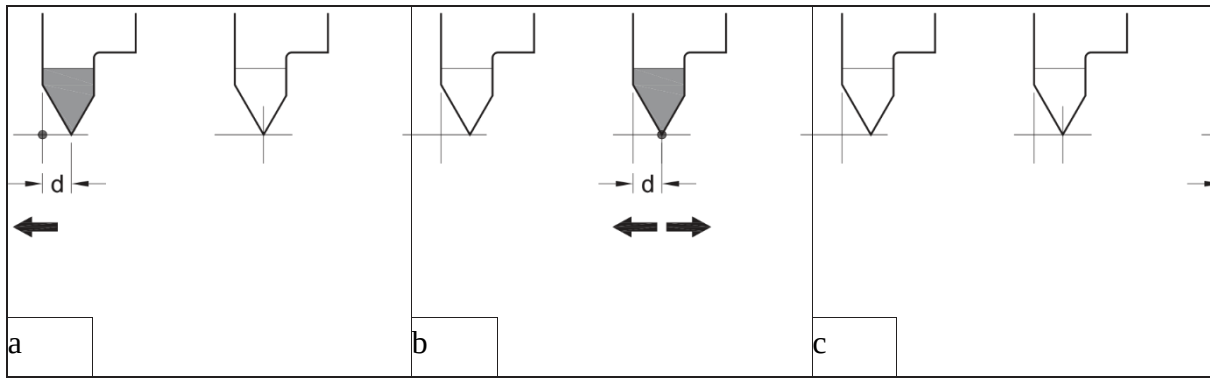


Figure 38-10

Typical reference points for setup of threading tools

Threading tool reference point selection requires more considerations than for turning tools. Figure 38-10 shows three possibilities, in the order of programming frequency.

The third version 'c' is the rarest and offers virtually no benefit to the programmer except in some cases of left hand threading. For most left hand threads, one of the first two versions is also quite sufficient.

Threading insert setting as in Figure 38-10a is the most suitable for general use and for threads that end at a shoulder. Configuration in Figure 38-10b is suitable for threads that end in an open diameter (recess). The Figure 38-10c shows a possible setting for a left hand threading work.

Selection of tool reference point (G50 or geometry offset setting) as per illustration in Figure 38-10a is the most desirable one, when the intent is to *standardize* tooling setup for *any* type of thread. It is the most convenient setting, regardless of how a thread ends. It is also the safest method, as it is measured closest to the part. In some cases, an allowance must be made for the difference between the programmed edge and the actual edge. Tooling catalogs list this value precisely, or one half of the threading insert width (if applicable) can be used instead as general rule.

F. BLOCK-BY-BLOCK THREADING

The oldest method of single point thread programming is to calculate each and every motion associated with the thread and write each motion as individual block of the program. This method is called *block-by-block* threading, long-hand threading, or just *block threading method*.

Each of the four basic motions occupies one block of program, resulting in the minimum of four blocks per each threading pass. If gradual pullout from the thread is used for thread cutting, there will be five blocks of program for each threading pass. When cutting coarse threads, threads in hard or exotic materials, even some multi start threads, this method often means quite a long program. The program length, difficulty in editing, high possibility of errors, and even small memory capacity of many older control systems, are negative sides of using this method.

On the positive side, CNC programmer has an *absolute programming control over the thread*. Such control placed into capable hands can often be applied to some special threading techniques, for example, cutting a thread shape with a threading tool much smaller than the thread itself or making large knuckle threads with a round grooving tool.

Thread programming method using the block technique for a constant lead thread is available on all CNC lathes that support threading.

G32	Thread cutting command (single point threading)

Preparatory command for this type of threading is G32. Command G33 may exist on some

controls, but G32 is the standard G-code for Fanuc and compatibles.

In an earlier example, a 3.0-12 TPI external thread was used. All cuts were distributed in six passes, for the total depth of 0.0511 - here is a summary:

Pass #1 depth	=	0.0210	Total depth	0.0210
			Diameter	2.9580
Pass #2 depth	=	0.0087	Total depth	0.0297
			Diameter	2.9406
Pass #3 depth	=	0.0067	Total depth	0.0364
			Diameter	2.9272
Pass #4 depth	=	0.0056	Total depth	0.0420
			Diameter	2.9160
Pass #5 depth	=	0.0050	Total depth	0.0470
			Diameter	2.9060
Pass #6 depth	=	0.0040	Total depth	0.0510
			Diameter	2.8980

Always make sure all diameters are calculated carefully without errors. Small error can cause big scraps.

The threading operation in program O3801 will use the tool and offset number 5 (T0505), at 450 r/min spindle speed (G97 S450) at normal spindle rotation:

O3801 (G32 VERSION - BASIC) ...

N59 T0500 M42

N60 G97 S450 M03

N61 G00 X3.3 Z0.25 T0505 M08 (START POSITION)

Now, the thread start point has been reached. The next stage is to implement all four steps, one step per block, for the *first* pass:

N62 X2.958 (PASS 1)

N63 G32 Z-1.6 F0.0833 (or F/E0.083333)

N64 G00 X3.3 N65 Z0.25

The remaining six passes can be programmed next, just by changing the diameters. Note that the threading feed rate does *not* repeat - it is modal from block N63 on.

N66 X2.9406 (PASS

N67 G32 Z-1.6 N68 G00 X3.3

N69 Z0.25

N70 X2.9272 (PASS

N71 G32 Z-1.6

N72 G00 X3.3

N73 Z0.25

N74 X2.916 (PASS

N75 G32 Z-1.6

N76 G00 X3.3

N77 Z0.25

N78 X2.906 (PASS

N79 G32 Z-1.6

N80 G00 X3.3

N81 Z0.25

N82 X2.898 (PASS 6)

N83 G32 Z-1.6

N84 G00 X3.3 N85 Z0.25

Block N85 terminates the threading routines, and the program can be closed as well, if there are no more tools used.

N86 X12.0 Z4.5 T0500 M09

N87 M30

%

What should strike as an odd occurrence in the example, is how many repetitions there are. Observe the three blocks following each new pass diameter - *they are always the same*. For a thread with many passes these repetitions will be numerous. This block-by-block method has one main benefit - is it under programmer's full control. Adjustments may be made to the number of threads and depth of each pass, but at the machine (at least not easily). Non-standard infeed method and a gradual pullout from the thread can also be added. Actual program editing after program has been completed is much more inconvenient.

G. BASIC THREADING CYCLE - G92

Computerized control systems can perform many internal calculations and store their results in control memory for further use. This feature is especially useful for thread cutting, since multiple repetitions of block-by-block tool motions can be avoided, and the program shortened quite significantly.

For better comparison and to illustrate a simple threading cycle, the same threading example that illustrated previous G32 command will be used again (12 TPI on a 3.000 inch external diameter), providing identical results. This cycle is usually called the G92 threading cycle on Fanuc or similar controls, also known as *box threading cycle*.

A word about *another* G92 command. Some programmers may be used to registering a current tool position with G92 command for milling applications. On CNC lathes, a command for the same purpose is G50, not G92. G92 used for threading has *nothing* to do with now virtually obsolete *position register* setting command. This applies for older control systems only - modern controls use advanced geometry offsets. Format for the G92 threading cycle is:

G92 X.. Z.. F..

+ where ...

X = Current threading pass diameter

Z = Thread end position

F = Threading feed rate in in/rev

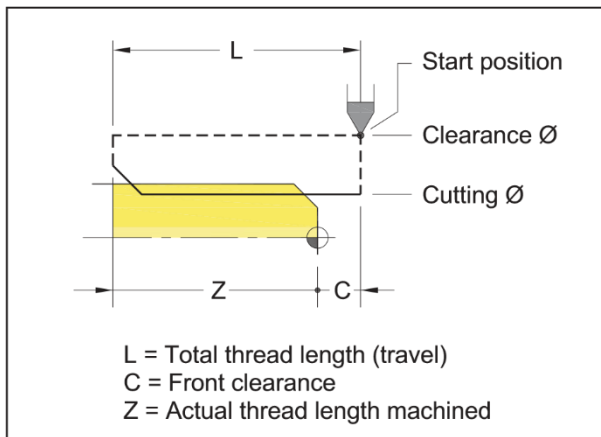


Figure 38-11

G92 - Simple thread cutting cycle, also called 'box threading cycle'

Schematic illustration of G92 straight thread cutting cycle is shown in Figure 38-11.

The program will do exactly the same job as with G32, except it will have a noticeably different structure.

Using G92 cycle, the following list shows all calculated diameters for each threading pass, as they will appear in the program (no change at this stage):

Pass #1 = 2.9580
Pass #2 = 2.9406
Pass #3 = 2.9272
Pass #4 = 2.9160
Pass #5 = 2.9060
Pass #6 = 2.8980

As before, the threading tool has been assigned a tool number and spindle speed - Tool 5 (T0505) and 450 r/min:

O3802 (G92 VERSION) ...

N59 T0500 M42

N60 G97 S450 M03

N61 G00 X3.3 Z0.25 T0505 M08 (START POSITION)

The first three blocks are identical to the block-by-block method of threading. Next, the threading tool will be positioned at the first pass diameter, cut the thread, retract and return to the starting position. The last three steps are always identical for each pass, and the main benefit of G92 threading cycle is that it eliminates such repetitive data and makes the program easier to edit.

G92 cycle will be called together with the first threading pass, in block N62. Note that X and Z axes input values and cutting feed rate are included in this block only:

N62 G92 X2.958 Z-1.6 F0.0833 (PASS 1)

Start position is defined by the last X and Z coordinate *before the cycle call*. In the example, start position is X3.3 Z0.25 (block N61). The remaining blocks include only data that have changed - which means only threading diameters. Five threading passes can be programmed just by

changing the X-diameters. There is no need to repeat the Z location or feed rate.

N63 X2.9406 (PASS

N64 X2.9272 (PASS

N65 X2.9160 (PASS

N66 X2.9060 (PASS

N67 X2.8980 (PASS

Block N67 is the last threading pass, followed by automatic return motion to the start position. From that position, program ends the same way as for G32.

N68 G00 X12.0 Z4.5 T0500 M09

N69 M30

%

One frequent programming mistake that can be made with this cycle is to omit G00 command in block N68. G92 cycle can be canceled only *by another motion command*, in this case by a rapid motion G00. If G00 is missing, the control system will expect that there are more threads to cut, which is not the case - undesired motions may happen.

Simple threading cycle G92 is just that - it is simple, without any frills. It does not have any special infeed methods, in fact, the only feeding method available is a straight plunge type (radial infeed). Later in this chapter, the plunge method of infeed will be described as *not* suitable for most threading operations. Automatic gradual pullout can be programmed with G92 by using M24 function prior to calling the G92 cycle, with examples later in this chapter.

Most controls support much more sophisticated threading method - also a cycle - called G76, described next.

H. MULTIPLE REPETITIVE CYCLE - G76

In *Chapter 35*, various lathe cycles were the main subject, normally used for turning and boring. In this section, a similar look will focus on another multiple repetitive cycles, this time used for various threading applications.

In early stages of CNC development, simple G92 threading cycle was a direct result of computerized technology of its time. Computer technology has been rapidly advancing and many great new features have been offered to CNC programmers. These new features simplify program development. One of the major additions is another lathe cycle, used for threading - a multiple repetitive threading cycle G76. This cycle is considered a complex cycle - not because it is difficult to use (on the contrary) but because it has some very powerful internal features.

To fully appreciate the impact of G76 threading cycle, compare it with the original G32 threading method, and even G92 cycle just described. While a program using the G32 method requires four or even five blocks for each threading pass, and G92 cycle requires one block for each threading pass, G76 cycle will do *any single thread* in *one* or *two* blocks of program, depending on control model. With G76 cycle, *any number* of threading passes will still occupy only a very small portion of the program, making any editing (if necessary) very easy and fast.

There are two programming formats available, depending on the control model. This is similar to programming of the other lathe cycles.

G76 Cycle Format - One Block Format

Any threading cycle requires initial data input - information provided to the control that defines a particular thread in machining terms. Figure 38-12 illustrates G76 cycle for older Fanuc 10/11/15T controls. Straight thread is shown.

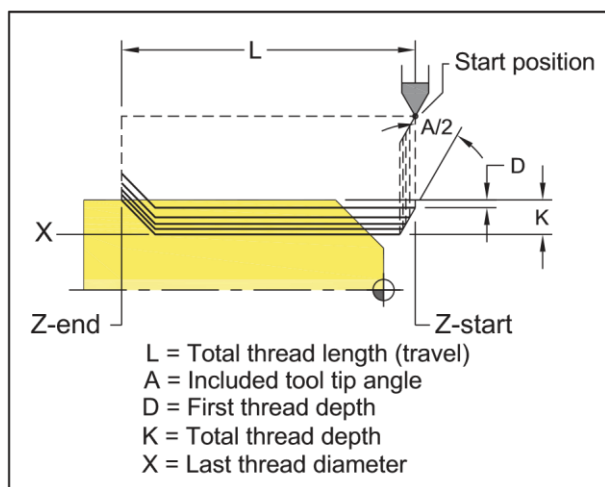


Figure 38-12

G76 multiple repetitive threading cycle for one block input - straight

The following parameters form the structure of G76 cycle applied as one-block (external or internal threads):

G76 X.. Z.. I.. K.. D.. A.. P.. F..

+ where ...

X	=	Last pass thread diameter (external or internal)
Z	=	Thread end position
I	=	Taper amount over total length (I0 for straight threads)
K	=	Actual thread depth per side - positive

D = First threading pass depth - positive (no dec. point) A = Included insert angle - positive (six selections)

P = Infeed method - positive (four selections)

F = Feed rate (always the same as thread lead)

Observe differences in the format structure for multiple repetitive cycle G76 with simple G92 cycle. G76 cycle appears to be simple as well, but the control system must do a large number of calculations and checks. All these calculations need data, in the form of input parameters that establish required thread specifications. Yet, in spite of more input parameters, G76 cycle is very easy to use in CNC thread programming.

G76 Cycle Format - Two Block Format

On the later Fanuc controls 0T, 16T, 18T, 21T and others, format of G76 cycle is somewhat changed from older models. Its purpose and function remain the same, the difference is only in the way how program data input is structured. Fanuc 10/11/15T use a single line cycle input, described earlier. Fanuc 0/16/18T/21 and other models require a two line input. Programmer does not have a choice - each format depends on the control system.

One-block and two-block G76 cycles are not interchangeable

If the control system requires a two-block entry for G76 cycle, programming format must cover two blocks:

G76 P.. Q.. R..

G76 X.. Z.. R.. P.. Q.. F..

+ where ...

FIRST block - starts with G76:

= ... is a *six-digit* data entry in *three pairs*:

Digits 1 and 2 - number of finish cuts (01-99)

Digits 3 and 4 - number of leads for gradual pull-out
(0.0-9.9 times lead), no decimal point used (00-99)

Digits 5 and 6 - angle of thread (tool tip angle)
(00, 29, 30, 55, 60, 80 degrees only)

= Minimum cutting depth (last depth of cut)
(Positive radial value - no decimal point)

= Fixed amount for finish allowance

(Decimal point allowed) SECOND block - also must start with G76:

X = Last thread pass diameter (absolute = X) ... *or* ...

Distance from the start position to the last thread diameter (incremental = U)

Z = Z-axis endpoint of thread (can also be incremental distance W)

R = Radial difference between start and end thread positions at the final pass
(R0 used for straight threads can be omitted)

P = Single depth of thread (height of thread)
(Positive radial value - no decimal point)

Q = First threading pass depth (largest cutting depth) (positive radial value - no decimal
int)

F = Feed rate (always the same as thread lead)

This format follows the logic of several lathe cycles described earlier in *Chapter 35*. Do not confuse P/Q letters of the first block with the P/Q letters of the second block. They all have their own meaning - within each block only! *Figure 38-13* shows some basic definitions of a straight two-block G76 threading cycle.

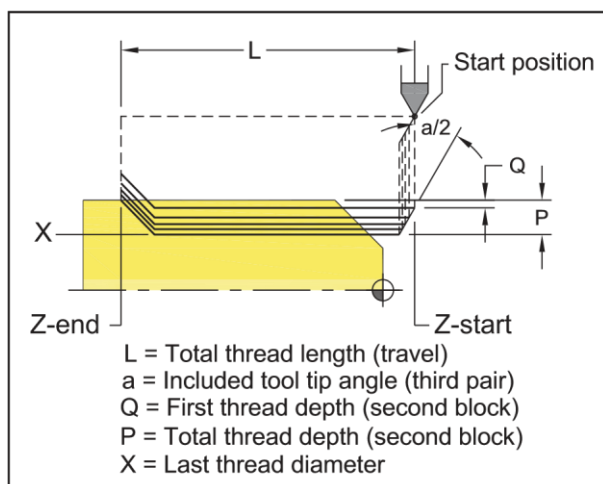


Figure 38-13

G76 multiple repetitive threading cycle for two block input - straight

One-Block vs Two-Block Format

Some programmers may ask why Fanuc has made such a dramatic change from a relatively simple single block entry to a more cumbersome double block entry. The answer can be one word - *flexibility*. Compare features that can now be programmed, features that did not exist in the single block format. For example, the *last depth of cut* (minimum cutting depth). In a single block format, this amount was only stored as a control system parameter and could be changed internally only. Now, the double block format allows the same parameter to be changed externally, through the program. Other features follow the same reasoning.

As both formats will be described in more detail in the next section, here are two typical and independent entries using a two-block G76 cycle:

Example - Metric units ... *Internal M761.5 thread*:

N10 G76 P011060 Q050 R0.05

N11 G76 X76.0 Z-30.0 P812 Q250 F1.5

Example - Imperial units ... *External 1-11/16 thread with 20 TPI*:

N20 G76 P011060 Q005 R0.003

N21 G76 X1.6261 Z-1.5 P0307 Q0100 F0.05

Programming Examples

The earlier threading example used for G32 and G92 was 3.0 external diameter with 12 TPI (*Figure 38-7, page 193*). It can be easily adapted to G76 programming method with a rather short output. Examples for both types of controls will be shown, using only the *minimum* number of program blocks (last tool shown in examples):

One-blockG76Cycle

O3803 (G76 VERSION - ONE BLOCK METHOD) ...

N59 T0500 M42

N60 G97 S450 M03

N61 G00 X3.3 Z0.25 T0505 M08 (START POSITION)

N62 G76 X2.8978 Z-1.6 I0 K0.0511 D0210 A60 P1 F0.0833

N63 G00 X12.0 Z4.5 T0500 M09

N64 M30

%

The fact that the entire threading section requires only six blocks is quite significant. Any programming change can be done by simple modification of selected data in block N62, which is the threading cycle. The most common change is the first threading pass depth, for example, from the current 0.0210 to 0.0180. All that has to be modified is address D0210 to D0180. As a result, more threading passes will be cut, presumably for a better quality thread. Feed rate can have up to six decimal places for long threads, and I0 can be omitted for straight threads.

Two-blockG76Cycle

O3804 (G76 VERSION - TWO BLOCK METHOD) ...

N59 T0500 M42

N60 G97 S450 M03

N61 G00 X3.3 Z0.25 T0505 M08 (START POSITION)

N62 G76 P011060 Q004 R0.002

N63 G76 X2.8978 Z-1.6 R0 P0511 Q0210 F0.0833

N64 G00 X12.0 Z4.5 T0500 M09

N65 M30

%

Although only one block longer than the older method, possible changes offer much more flexibility at the machine. No more changing system parameters - several more options are now quickly available, if needed.

The comparison of the G76 cycle with G92 cycle is unfair, as each cycle is product of a different technological era. They coexist in the same control unit even now, mainly to be *downward compatible* with older programs. These two cycles provide good illustration of some significant differences between programming techniques.

For example, in the G92 threading cycle application, input of *each* thread pass diameter is important, in G76 cycle, only the *last* pass diameter input is important.

First Thread Diameter Calculation

Control system calculates the actual first pass diameter in a very similar way as done manually in G32 or G92 threading modes. Calculation of the *first* threading diameter is based on supplied *external* thread information:

Root diameter

[X address]

Total thread depth

[K or P address]

First thread depth

[D or Q address]

For *one block* format, and based on supplied values, the first threading diameter T_f will be calculated as:

T_f	X	K	2	D	2

In the example, X is 2.8978, K is 0.0511, the first threading depth D is 0.0210, entered in the program as D0210. Therefore, the first pass threading diameter T_f will be:

$$T_f = 2.8978 + 0.0511 \cdot 2 - 0.021 \cdot 2 \quad T_f = 2.9580$$

For *two block* format, and based on supplied values, the first threading diameter T_f will be calculated as:

T_f	X	P	2	Q	2

In the example, X is 2.8978, P is 0511, the first threading depth Q is 0.021, entered in the program as Q0210. Therefore, the first pass threading diameter T_f will be:

$$T_f = 2.8978 + 0.0511 \cdot 2 - 0.021 \cdot 2 \quad T_f = 2.9580$$

Both P and Q of the two block example are amounts entered in the second G76 block. In either case, the result is the same diameter as in previous simpler examples, but this time it was calculated by the control unit. Both formulas can be adapted to calculate the first threading diameter for internal threading:

T_f	X	K	2	D	2

T_f	X	P	2	Q	2

For everyday programming, the first diameter calculation is not necessary for either G76 cycle or is included here only to explain how the control system calculates it.

Note that the total thread depth uses 0.541 constant for internal threads rather than the 0.613 constant for external threads.

I. THREAD INFEEED METHODS

Entry of a threading tool into material can be programmed in more than one way. One of the most important selections is a method that controls threading tool approach towards the thread, also known as *threading infeed*. This is a method detailing threading tool motions, using one of two basic methods of infeed, as illustrated in *Figure 38-14*.

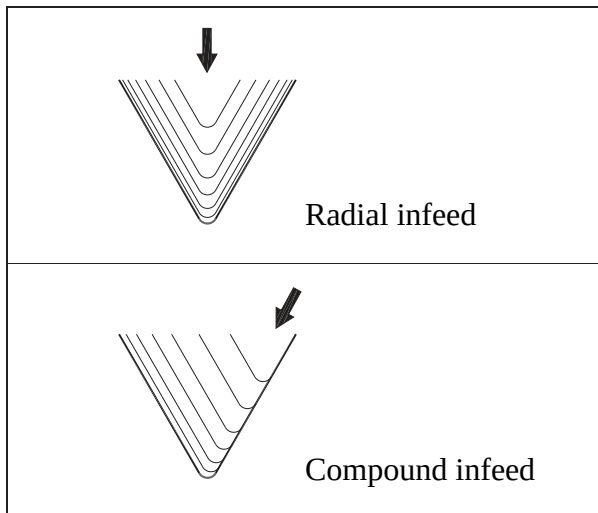


Figure 38-14

Radial and compound infeed for thread cutting

The simplest infeed method in thread programming is called the *radial* infeed method, also known as the *plunge*, *straight*, or *perpendicular* method. More common method is called an *angular* method, better known as a *compound* infeed or a *flank* infeed.

A need to control infeed direction in threading is to offer the best cutting conditions for cutting tool edge. Except for threads with very fine leads and some soft materials, the majority of threading cuts will benefit from a compound infeed (at an angle). Some threaded shapes are excluded for the reason of their geometry - for example, a square thread will always need a radial infeed (straight plunge infeed). The angle of infeed is programmed as the included tool tip angle in both types of G76 cycle.

Each infeed method has its variations, using the following features for one purpose - *the best chip load control*:

Constant cutting amount

Constant cutting depth

One edge cutting

Both edges cutting

There is a lesser known setting P in the one-block G76 cycle that selects each procedure - see page 368.

Radial Infeed (Straight Infeed)

Radial infeed method is a common threading method for some jobs. For horizontal threads, it is used when a perpendicular motion towards the spindle centerline is required.

In G76 one block threading cycle, **A0** parameter is used for a radial infeed. In a two block threading cycle, the last two digits of the first block P-address will be two zeros, as in **P----00**.

In G32 block-by-block programming and G92 simple threading cycle, there is no parameter to program. The Z axis start position is the same for all thread diameters and is easier to program. The radial infeed is suitable for soft materials (brass, some aluminum, etc.), but it could damage threads cut in harder metals.

Main result of a radial infeed motion is that *both* insert cutting edges are removing material at the *same time*. Since both edges are opposite to each other, curling of the chips will also be opposite to each other. In many applications, this will cause high temperature and tool wear problems related to heat and may create a groove-like channel on the insert. Even decreasing depth for each infeed may not eliminate the problem. If radial infeed does not produce a high quality thread, a compound infeed approach will generally do a much better job.

Compound Infeed (Flank Infeed)

Compound infeed method, also called a *flank* method, uses an *angular* cutting direction that moves pass bypass towards the final thread depth. Chip shape produced by compound threading method is similar to the shape of a chip produced by turning. Only one insert edge does the actual cutting, so most generated heat dissipates *away* from tool edge, while chips curl away, extending tool life. The chip depth can be heavier and fewer passes will be required for most threads. *Figure 38-14* shows compound infeed, where one cutting edge is in constant contact with a thread side. There is no cutting, only undesirable rubbing which may cause a poor surface finish and shorten tool life. To avoid this problem, calculate each threading pass position with a small angular clearance - *Figure 38-15*.

For example, a typical V-thread, with 60° included angle has a flank angle of 30° - in this case, selected infeed angle should be a little less than that, say 29, which provides a 1 clearance on one thread side. *Keep in mind* - the thread shape or geometry is not changed, as it is built into the physical shape of cutting insert. What is changing is the way *how* the insert will cut. An example using G32 threading method with modified compound infeed will be provided in the next section.

G76 threading cycle offers some very powerful input options, in forms of cutting parameters. There is no selection of infeed method available directly, but infeed can be influenced by other settings, particularly the tool tip angle. This setting is available for both types of G76 cycle.

Thread Insert Angle

For compound infeed, a *non-zero* value is assigned to the parameter that represents tool angle, a value equal to the *included* angle of threading insert. Only the following six angle settings are allowed in a one-block G76 threading cycle:

A0	A29	A30	A55	A60	A80
A0	Radial (straight) infeed			ISO	
A29	ACME type of thread			ANSI	
A30	Metric Trapezoidal / Round thread			DIN 103 / DIN 405	
A55	Whitworth 55 thread			BSW, BSF, BSP	
A60	Standard 60 V-thread			M, UN, UNJ	
A80	German PG thread			Panserrohrgewinde	

For a two-block G76 cycle, the same six options are programmed as the last two digits of the *first* block P-address:

G76 P----00 Q.. R..

G76 P----29 Q.. R..

G76 P----30 Q.. R..

G76 P----55 Q.. R..

G76 P----60 Q.. R..

G76 P----80 Q.. R..

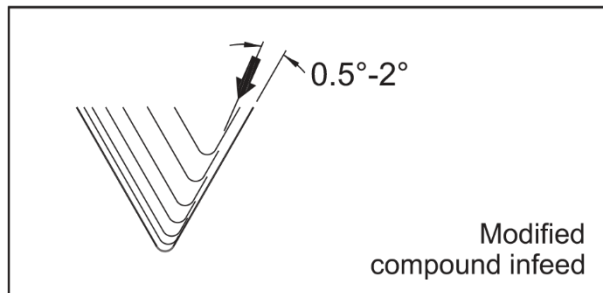


Figure 38-15

Modified compound infeed angle for better thread quality

In theory, *any* of the six available options can be used for *any* thread, regardless of its actual tool tip angle. This may offer some creativity to find the 'best' infeed method.

In practical applications, *ACME* thread (29 included angle) is very common for transmission of motion, for example a tire changing jack. *Metric Trapezoid* thread is the metric version of *ACME* thread and has a 30 included angle. *Whitworth* thread has an included angle of 55, with origin in Great Britain; it has limited usage, as metric threads become standard worldwide. *PG* thread is a special German pipe thread (*Panßerrohrgewinde*), with 80 angle, not common in North America - it looks like a garden hose thread.

One-block G76 cycle (for older controls) offers a *P* parameter, that works very closely with the *A* parameter and defines the *thread cutting type*.

Thread Cutting Type - Address P

In a one-block G76 threading cycle, threading infeed can be programmed with the address *P*, in addition to the address *A*. The purpose of threading parameter *A* is to control the threading infeed method - up to a certain extent - based on the *included angle* of threading insert. For a more controlled infeed method, a method that controls the threading *depth*, there is also parameter *P*, programmable in the G76 cycle format and available for the Fanuc controls that use one-block G76 input. It defines the *thread cutting type*, relating to the programmed thread depth.

In addition to radial infeed (straight or plunge), programmed with *A0* parameter and compound infeed - a non-zero parameter *A* - there are two other main cutting types that can be used in programming a thread infeed - a *one side cut* and a *zigzag cut*. These terms refer to the number of cutting edges employed at one time. The one side cut refers to cutting with *one edge*, the zigzag cut refers to cutting with *two cutting edges*. Each of them can be used in conjunction with the selected *A* thread angle parameter and cutting depth - either as a *constant amount* or a *constant depth*.

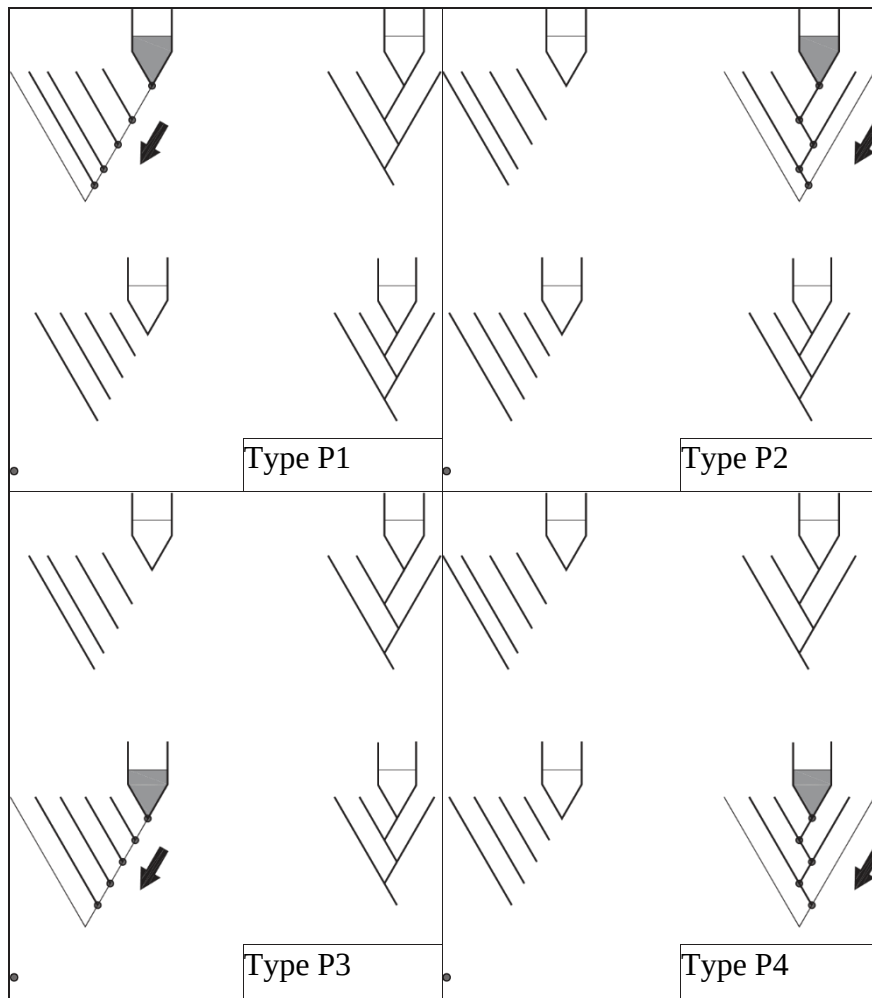


Figure 38-16

Cutting types for one-block G76 threading cycle

'P' parameter

Fanuc CNC lathe controls offer four methods of controlling the thread cutting depth infeed (Figure 38-16):

P1 One edge cutting ... with constant cutting amount

P2 Two edges cutting ... with constant cutting amount

P3 One edge cutting ... with constant cutting depth P4 Two edges cutting ... with constant cutting depth On Fanuc lathe controls manufactured before model 10T, the P parameter in G76 cycle was not available. The equivalent of what is now the P1 parameter was the default. On controls that *do* support P parameter, if the P number is omitted in G76 cycle call, P1 cutting method is assumed as a default. This is the most common threading application and will be suitable for many jobs. It will apply *one cutting edge* of the threading tool and the *constant cutting amount*. That will result in equal chip volume removal. Feel free to experiment with the other three options as well.

J. COMPOUND INFEEED CALCULATIONS

Compound (flank) infeed does not present programming problems when using the advanced G76 threading cycle. If a CNC system has G76 cycle available, it can be used for about 95% of all work. What about the remaining 5%? What if G76 cycle cannot be used and a program needs fully controlled compound infeed? How to control other infeed methods available for G76, without having the G76 cycle available or finding it impractical to use?

Unfortunately, there is only one way - take a pocket calculator and calculate each and every tool position and tool motion individually. Is it a lot of work? Yes. Is it worth doing? Absolutely. It has to be a really good job from the start, because even a slight modification at the machine could be very difficult. Atop class programming job is always worth the extra time and effort when quality and precision of the final part depend on it. Quality is not instant - programmers (and machine operators) have to invest some extra work and time into it.

General principles of compound threading as applied to a block-by-block programming are simple, but the programming work may be tedious and editing on the machine may be impractical. Each threading pass has to be calculated in a different Z-axis start position. This is called the *shifted* position that must be calculated exactly, otherwise the program will fail. It also had better be right the first time, otherwise any changes could be long and costly. Again, in this example, the same thread will be used as in previous examples (3.0-12 TPI external thread, introduced earlier, on *page 355*). Program will use G32 threading command, with a modified compound infeed at 29.

Initial Considerations

Thread used for the examples in this section is a 3.0-12 TPI external thread. All individual diameters for each threading pass had been calculated earlier, and all single depths for each pass had been established as well at the same time. These values will be used in this example as well. In total, there are six diameters and six depths to program the Z-value (total depth accumulation is 0.0511 per side), as illustrated in *Figure 38-17*.

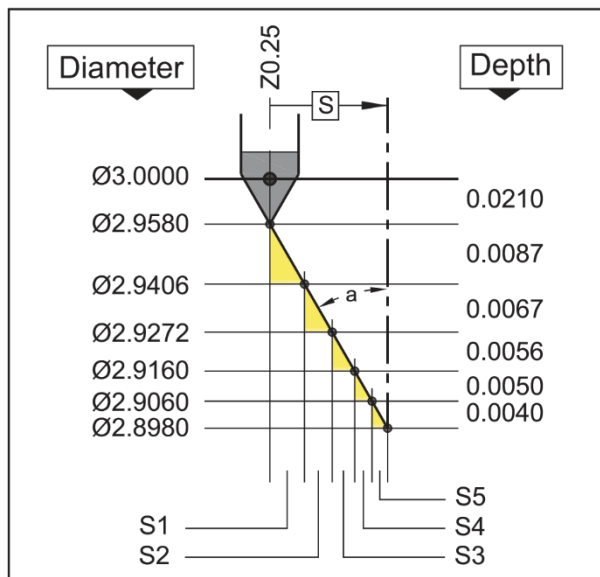


Figure 38-17

Calculations required for compound infeed used in G32 threading mode

The illustration shows distribution of each single threading depth for all six threading passes and matches the following summary:

Pass #1	single depth of 0.0210	at 2.9580
Pass #2	single depth of 0.0087	at 2.9406
Pass #3	single depth of 0.0067	at 2.9272
Pass #4	single depth of 0.0056	at 2.9160
Pass #5	single depth of 0.0050	at 2.9060
Pass #6	single depth of 0.0040	at 2.8980

In addition to individual diameters and depths, *Figure 38-17* also shows combined *shift* of individual S1-S5 shift amounts as the S amount. When Z-start position is shifted, the shift distance

must be calculated on the basis of compound angle *and* the threading pass single depth, but only for the *remaining* passes. That means for six threading passes, there will be five shifts. Every new calculation must always be based on the last calculation.

Z-axis Start Position Calculation

Illustrated distance *S* represents the *total* shift amount from the nominal Z-axis start position - in the example, this position is at Z0.25. Usually, the shift is to the Z-positive direction, mainly for reasons of increasing the clearance rather than decreasing it. Opposite direction is possible.

Theoretically, it makes no difference *which* direction the shift is programmed - *towards* the thread or *away from* the thread. Practically, it is always better to program the shift *away from* the thread, if possible - the tailstock may be in the way, so watch its position. This way, the distance for the feed rate acceleration will *increase*, rather than decrease.

Although another approach may also be chosen, the *S* distance will be calculated first, mainly for reference purposes. Only five shifts are required, so the total shift *S* must be calculated *after* the first pass is completed. It means from the first depth 2.958 to final depth 2.898:

$$(2.958 - 2.898) / 2 = 0.0300$$

Total thread depth after the first pass is 0.03 and the selected compound infeed angle *a* is 29°, so using a standard trigonometric formula will provide the *S* distance based on infeed angle *a*:

$$S = 0.03 \tan 29 = 0.016629272 = 0.0166$$

The *S* distance represents the *total* shift of a threading tool from the start position. Shift for each threading pass will be its relative share of the *S* value. Each share is identified as an *S_n* in the *Figure 38-17*, within 5 passes - *S*1 to *S*5.

<i>S_n</i>	<i>D</i>	$\tan 29$

+ where ...

S_n = Current thread pass shift - incremental amount *n* = Current pass number
(shift count) *D* = Single depth of current thread pass

Calculation for each pass uses the same basic formula, changing the *D* depth input. Keep in mind that the purpose of this process is to find a new Z-axis start position for *each threading pass* - i.e., the Z shift amount for a given thread diameter. *Figure 38-18* illustrates the calculation.

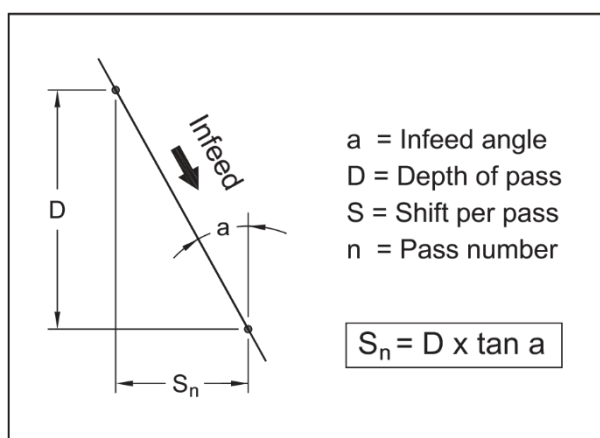


Figure 38-18

Calculation of thread Z-start position for individual passes

Once the Z-axis start position is established (Z0.25), it is used for the remaining five shifts. *There is no shift for the first pass.* The remaining five shifts are based on the initial position and can be calculated from the above formula:

S1 = 0.0087 tan29 = 0.0048

S2 = 0.0067 tan29 = 0.0037

S3 = 0.0056 tan29 = 0.0031

S4 = 0.0050 tan29 = 0.0028

S5 = 0.0040 tan29 = 0.0022

T O T A L = 0.0166

The five shifted positions for Z-axis start position can be calculated, based on the selected start position of Z0.25:

Pass #1 = Z0.2500 += Z0.25

Pass #2 = Z0.2500 += Z0.2548
048

Pass #3 = Z0.2548 += Z0.2585
037

Pass #4 = Z0.2585 += Z0.2616
031

Pass #5 = Z0.2616 += Z0.2644
028

Pass #6 = Z0.2644 += Z0.2666
022

This example shows progressive shift from the initial position of Z0.25, *away* from the thread. Using this method offers a certain level of confidence that the original Z0.25 minimum clearance will never be smaller. Only the Z-axis value will change - other programmed values are not affected by this programming method at all.

The final program is not short, even for only six passes, which is typical with G32 programming, but it does illustrate the compound method of threading when no cycle is available or is not practical to use. Only the threading tool is shown in the example O3805.

O3805 (G32 THREADING - COMPOUND INFED) ...

N59 T0500 M42

N60 G97 S450 M03

N61 G00 X3.3 Z0.25 T0505 M08 (START POSITION)

N62 X2.958 (PASS 1)

N63 G32 Z-1.6 F0.0833 (or F/E0.083333) N64 G00 X3.3

N65 Z0.2548 (SHIFT S1)

N66 X2.9406 (PASS 2)

N67 G32 Z-1.6

N68 G00 X3.3

N69 Z0.2585 (SHIFT S2)

N70 X2.9272 (PASS 3)
N71 G32 Z-1.6
N72 G00 X3.3
N73 Z0.2616 (SHIFT S3)

N74 X2.916 (PASS 4)
N75 G32 Z-1.6 N76 G00
.3
N77 Z0.2644 (SHIFT S4)

N78 X2.906 (PASS 5)
N79 G32 Z-1.6
N80 G00 X3.3
N81 Z0.2666 (SHIFT S5)

N82 X2.898 (PASS 6)
N83 G32 Z-1.6
N84 G00 X3.3
N85 Z0.25 M09
N86 X12.0 Z4.5 T0500
N87 M30
%

In program O3805, the thread infeed method is equivalent to the P1 parameter in G76 cycle (one-block method). This cutting type employs only a single edge of the threading insert, with a constant amount per each threading pass. It represents the most common programming method for threads and can be used as a sample for many other thread cutting applications. Block-by-block threads will be longer and will need to be checked for accuracy very carefully.

K. THREAD RETRACT MOTION

As stated already, there are only two methods of retracting threading tool from the part - a straight motion along *a single axis*, and a gradual *simultaneous* motion along *two axes*. Both are used in single point thread programming. In fact, their frequent applications even justify special methods of program input.

The first method has actually been described already see *page 364* (G76 - first block). G76 cycle in its two-block format provides an input parameter that allows setting of the retract motion - the *gradual pull-out* from the thread. In this case, there is no need for any other programming interference - this feature is built in the two-block G76 cycle.

For older - and still very common controls - using the *one block* G76 entry, there is a different solution. There are two miscellaneous functions built into the control system as a standard feature. These threads retract functions are called the *thread chamfering functions* or *thread finishing functions*. Another description is a bit more accurate (or at least descriptive) than these two - *thread*

pullout functions.

Thread Pullout Functions

When using threading cycle G76 for CNC lathe work, the thread end position (Z-axis value) will either be in *solid* material or in an *open* space (a *groove recess*, for example, or an *undercut*). Programmed pullout can be programmed along a single axis, or along both axes simultaneously.

For older controls, typical Fanuc functions designed for this purpose are M23 and M24. They control the pullout of the threading tool at the thread end:

M23	Thread finishing ON	(two axes pull-out)
M24	Thread finishing OFF	(One axis pull-out)

Other machine controls may have similar functions. The purpose of these functions is to enable or disable the *automatic insertion* of a pullout retract between threading motion sequences 2 and 3, as described earlier in this chapter. *Figure 38-19* illustrates the comparison of a threading motion *with* and *without* gradual pullout for both methods.

M23 and M24 functions apply only to G76 one block method

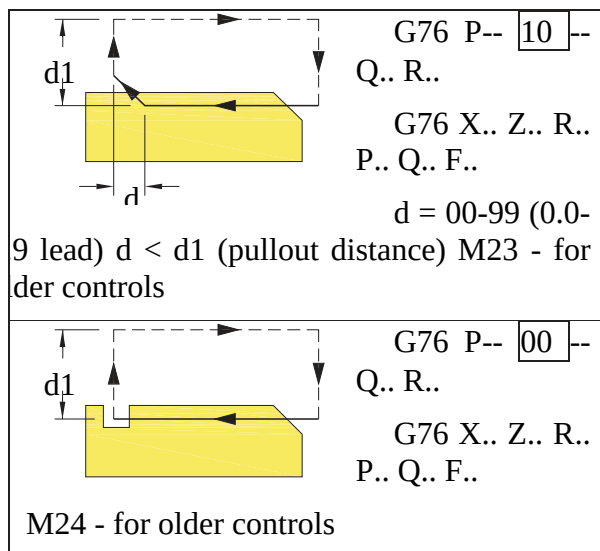


Figure 38-19

Typical gradual pull-off from thread - 1.0 lead shown in upper image Only older controls require M-functions M23/M24

Single Axis Pullout

A single axis pullout (thread finishing OFF) is a simple rapid motion programmed at the end of threading pass as the *third motion* of the four threading sequences. Pullout direction is always in X-axis. For either threading cycle G92 or G76 (one-block method), this is the default condition, so M24 is not needed, unless M23 function is used as well, usually for another thread in the same program. These two functions cancel each other. If M24 function is used, it must be programmed *before* the threading cycle for which it has been applied. For example, threading program O3803 using the G76 one-block cycle will be slightly modified in O3806:

O3806 (G76 VERSION - ONE BLOCK METHOD) ...

N58 M24 (THREAD PULLOUT OFF)

—

N59 T0500 M42

N60 G97 S450 M03

N61 G00 X3.3 Z0.25 T0505 M08 (START POSITION)
N62 G76 X2.8978 Z-1.6 I0 K0.0511 D0210 A60 P1 F0.0833
N63 G00 X12.0 Z4.5 T0500 M09
N64 M30

%

M24 function appears in block N58, the only block that was available without another M function. Example for a single axis pullout will be a modified program O3804:

O3807 (G76 VERSION - TWO BLOCK METHOD) ...

N59 T0500 M42
N60 G97 S450 M03
N61 G00 X3.3 Z0.25 T0505 M08 (START POSITION) N62 G76 P010060 Q004 R0.002

—

N63 G76 X2.8978 Z-1.6 R0 P0511 Q0210 F0.0833
N64 G00 X12.0 Z4.5 T0500 M09
N65 M30

%

Underlined item in N62 has been changed from 10 to 00.

Two-Axis Pullout

Two-axis pullout is a gradual angular tool motion along two axes, away from the thread (thread finishing ON). The example O3808 is similar to the previous example:

O3808 (G76 VERSION - ONE BLOCK METHOD) ...

N58 M23 (THREAD PULLOUT ON)

—

N59 T0500 M42
N60 G97 S450 M03
N61 G00 X3.3 Z0.25 T0505 M08 (START POSITION)
N62 G76 X2.8978 Z-1.6 I0 K0.0511 D0210 A60 P1 F0.0833
N63 G00 X12.0 Z4.5 T0500 M09
N64 M24 (CANCEL M23)

—

N65 M30
 %

In this case, M23 was applied in block N58 and an additional block N64 was used to cancel the pullout. This cancellation was not necessary in this program, but it is a good practice to cancel functions used only for specific purposes.

There are some conditions that apply to the M23 function. In *Figure 38-19*, the finishing distance d is set by the control parameter, within the range of 0.100 to 12.700 the thread lead. Normal control setting is equivalent to *one time* the thread lead ($d=1.0$). Pullout angle from the thread is usually at 45, or a little less because of a delay in the servo system. If the finishing distance d is greater than the pullout distance $d1$, the pullout will *not* be done.

Program example O3804 on *page 365* shows a pullout of one lead (1.0 = 10) in block N62:

N62 G76 P011060 Q004 R0.002

—

N63 G76 X2.8978 Z-1.6 R0 P0511 Q02100 F0.0833

In the G76 first block, the address *P* has six digits, separated into three pairs. Do not confuse the first pair with the second pair - they both indicate the amount of *one* with totally different meanings:

P01xxxx indicates *one finishing cut* within the range of 00 to 99 integers only.

Pxx10xx indicates *one lead* for pullout, within the range of 0.1 to 9.0. Real numbers are allowed, but the decimal point is not written.

L. HAND OF THREAD

Any thread can be cut in either *right hand* or *left hand* orientation. Neither selection has any effect on the profile or thread depth, but other factors are important. The majority of threading applications use the right hand thread. *Right hand* and *left hand* terms relate to the helix of the thread *Figure 38-20*.

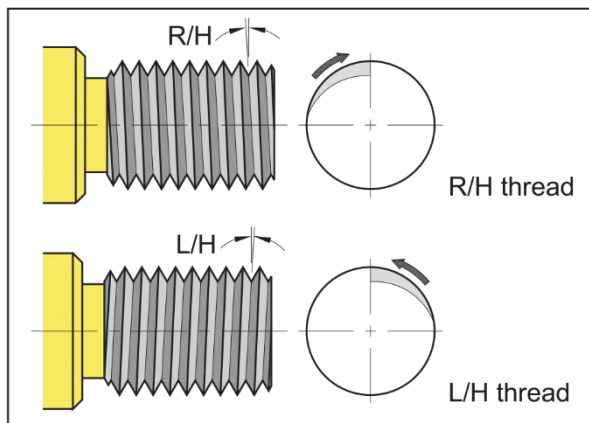


Figure 38-20

Right Hand and Left Hand threads shown on a screw

In CNC programming, the hand of thread is determined by two conditions:

- Cutting direction of the tool (Z+ or Z-)
- Direction of the spindle rotation (M03 or M04)

These conditions are used in various combinations in order to program a particular thread. Factors that influence the programming method for a R/H and L/H thread are:

- Threading tool design - right hand or left hand
- External or Internal thread
- Front or Rear lathe design
- Spindle rotation direction - M03 or M04
- Cutting direction - Z+ or Z-
- Tool tip orientation in the turret

Theoretically, either hand of thread can be cut with *any* threading tool, but this approach is not right. A poor choice affects the thread quality, life of threading insert, additional costs are involved, etc. When a thread starts too close to a shoulder (in a recess), the clearance for acceleration is limited. The only method to prevent imperfect threads due to acceleration in a small area is to *decrease* the spindle speed.

◆ Configurations for Hand of Thread

Often, it is difficult to visualize the actual machining, and making a mistake is quite easy. The most important consideration is that *all* factors involved are correct - even one incorrect factor will make a wrong thread.

Tooling manufacturers recognize this situation and provide graphical method similar to the ones shown here. All eight combinations cover R/H and L/H external and internal threads for a rear type lathe. Design of tool may influence programming - tools are shown at their start position - M03/M04 spindle rotation is described on *page 82*.

Figures 38-21, 38-22, 38-23, 38-24

Hand of external threads - REAR lathe applications

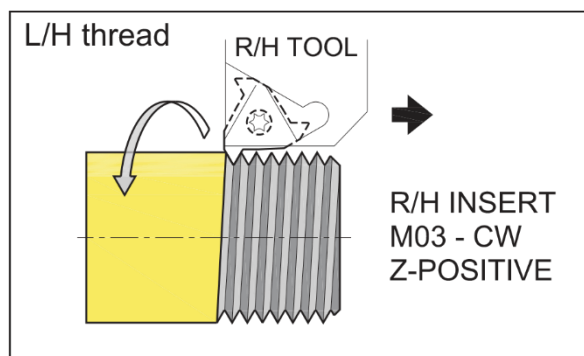
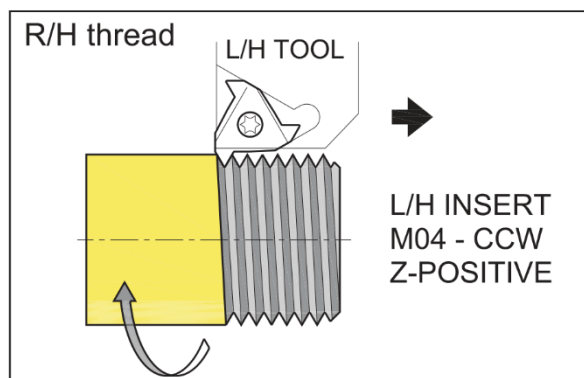
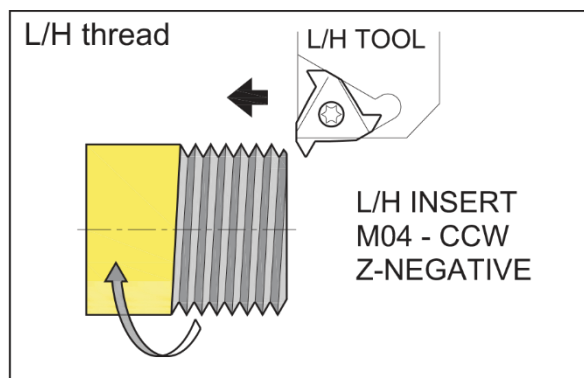
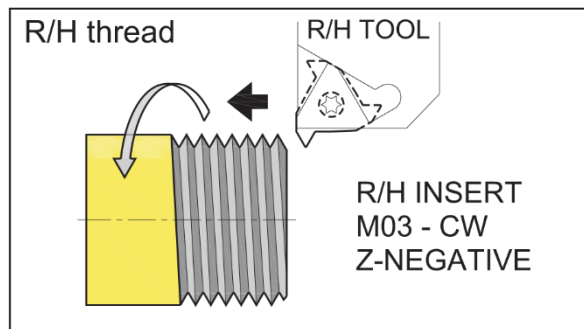
R/H thread

R/H BAR

R/H INSERT

M03 - CW

Z-NEGATIVE



L/H thread

L/H BAR

L/H INSERT

M04 - CCW

Z-NEGATIVE

R/H thread

L/H BAR

L/H INSERT

M04 - CCW

Z-POSITIVE

L/H thread

R/H BAR

R/H INSERT M03 - CW

Z-POSITIVE

Figures 38-25, 38-26, 38-27, 38-28

Hand of internal threads - REAR lathe applications

M. THREADING TO A SHOULDER

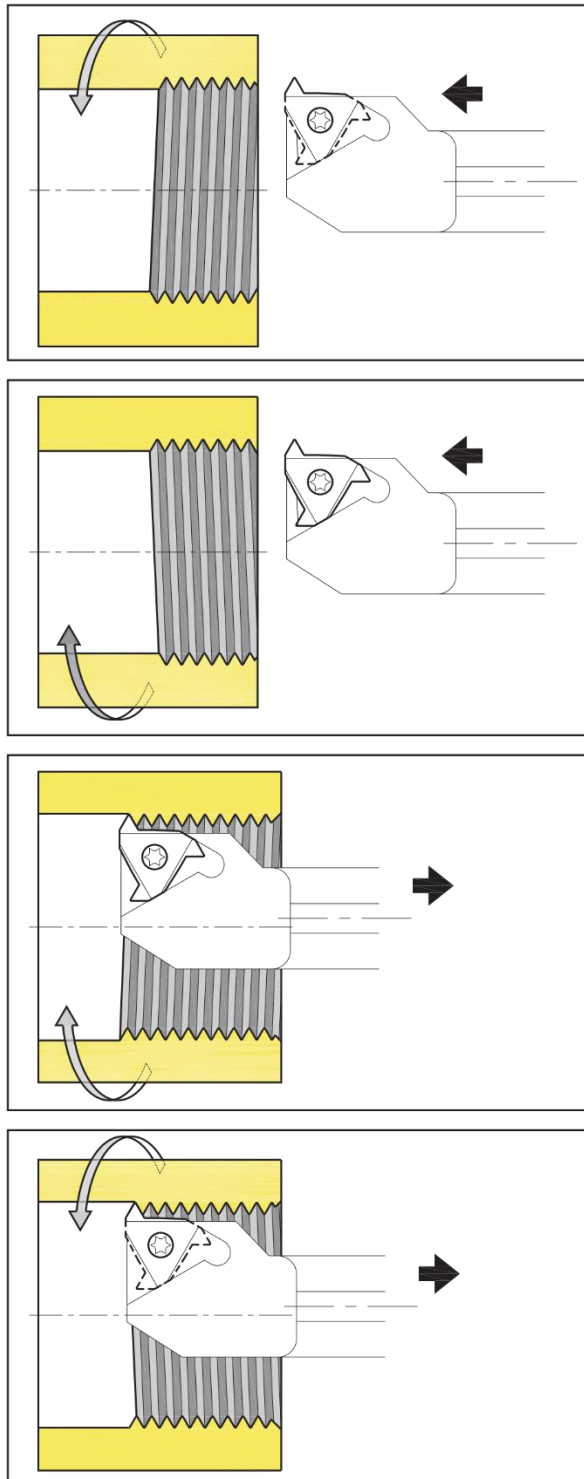
Programming a thread that terminates at a shoulder presents a unique difficulty. This difficulty is a wall - better known as *shoulder* of the part. It is not enough to program the thread end point reasonably - it must be programmed *exactly*. Even then, a collision is possible if the tool setup is not accurate. There are three typical problems in this area of thread programming:

Recess groove is too narrow or non-existent

Threading insert is too wide ^{Thread} is too deep

The first problem of threading towards a shoulder, a narrow width of the recess groove, is easy to correct - just increase recess width in the program. The majority of recess grooves can be adjusted for threading tool, without damaging engineering intent behind the design. This may be a justified case of 'overruling' the drawing - but check first anyway!

The second and third problems may not be related, but their solution is usually the same. If the threading insert is too wide or thread is too deep, try to increase the recess width first, if at all possible. If the recess width cannot be increased, for whatever reason, then there is another choice - to *decrease* the threading insert width. One obvious solution is to change the current threading tool for a smaller one that can still cut the required thread depth. This may be an insert one size smaller, which usually requires a different tool holder as well.



If a smaller tool cannot be used, program for a *modified existing threading insert*. Modification in this case means physical grinding off the problematic portion of the insert, without disturbing the portion that actually removes material. Before deciding on any modification by grinding, consider other options carefully - altering standard tools designed for CNC work should always be the *last* resort, not an automatic first choice. A coated insert will lose its cutting advantages, if the coating is removed by grinding. Be careful not to grind off coating within the cutting section of the insert. In case the part program *does* use a modified threading insert, a few suggestions may help to do it with more knowledge.

Always use care with modified tools

Insert Modification

There is a number of standard threading inserts in every tooling catalogue and chances of

finding one suitable for the job at hand are very good. In case a standard threading insert needs modification, the following example illustrates a few programming methods - incidentally, it is irrelevant if there is or there is *not* a recess groove on the part.

To modify a standard threading insert, look at its normal configuration first. *Figure 38-29* shows a typical threading insert with the known width W and the angular length A , tip radius or flat R , and an unknown angular height H .

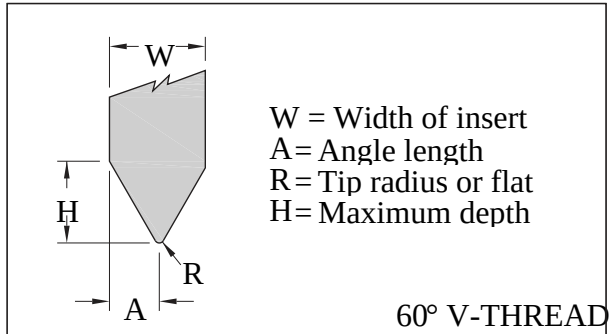


Figure 38-29

Essential dimensions of a typical threading insert

In the example, catalogue based insert dimension shown as W is 0.250 and A dimension is 0.130. Included angle of the threading insert is 60° and insert flat or tip radius R is 0.012, which is not relevant in this case. Dimension H indicates the maximum theoretical thread depth and is normally measured to the sharp point of insert tip. It is calculated using a trigonometric function:

$$H = A / \tan 30$$

$$H = 0.130 / 0.577350269$$

$$H = 0.225166605$$

$$H = 0.2252$$

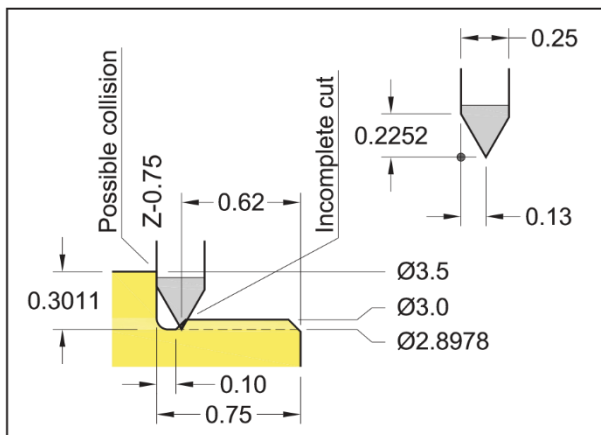


Figure 38-30

Threading insert before modification may cause problems

Two possible threading problems are illustrated above, in *Figure 38-30*. The example shows programming a thread with a 0.100 recess groove width, using an insert with angular length A of 0.130. This insert is *not* suitable for the job, as it cannot provide the minimum full depth thread length - the difference between the shoulder length and recess width:

$$0.750 - 0.100 = 0.650$$

The minimum thread length shown is only 0.620. There is no clearance, and the thread length is too short. To solve this problem, select a smaller size threading insert if possible. If not,

modification of a larger insert is the only way.

Such modification requires grinding of the insert in noncritical areas, to allow it to complete the minimum 0.650 thread length. In theory, the minimum amount to be ground off the insert is 0.030, difference between the required and actual thread lengths. This modification does not provide any *clearance* at the thread shoulder or at insert tip. These two clearances are essential for the best threading results. Even a minor setup error at the machine can cause some serious difficulty.

Always calculate modification amounts, never guess them

In the example, there are three dimensions that influence the actual amount of insert modification. Summing up all three will be the amount to be ground off the insert. One, the thread length has to be extended by 0.030 to achieve 0.650 minimum length. Two, clearance from the shoulder will also be 0.030, and three, clearance past the thread end will be 0.020. The last two clearances are arbitrary decisions by the programmer to suit the job. Final result is the total amount of insert modification as 0.080. It means the amount of 0.080 must be ground off the original larger threading insert. That will shorten the original angular length of 0.130 to the length of 0.050. Always make sure full depth of thread can be achieved with the modified insert. Part program will reflect the modification in Z-axis thread end position, which will be written as Z-0.8. Setup position of the insert - its *command point* - does not change. See *Figure 38-31* for complete solution.

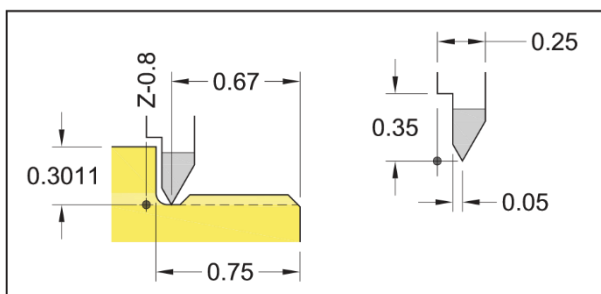


Figure 38-31

Modified threading insert provides sufficient clearance in the recess

In threading, thread length is *the actual length of the full depth thread*. Part design often allows a little longer thread, but not shorter. The shoulder height is also important. In the example, shoulder is only 0.3011 high, so insert modification was possible. A large threading insert may not always be modified, and the only solution will be to use a smaller insert size.

Program Testing

Whether a threading insert used is based on catalogue dimensions or a modified insert, threading to a shoulder presents a time of special caution by the CNC operator, when the first part is produced. Since feed rate override and Feed hold switches are disabled during threading, program verification on the lathe will become more difficult. Even computer based graphic testing toolpath methods may not show all possible collisions.

A simple, yet very effective, thread program checking method is always available, right at the CNC lathe. This method requires a skilled CNC lathe operator, who does understand both the program and threading principles well. Knowledge of machine operation is also important.

This method employs several features found on contemporary CNC controls. Main purpose of the program test is to find out if the threading tool will collide with the shoulder *before* actual threading cut takes place.

The following steps are only general - adapt them to suit actual conditions when testing a thread cutting program:

Use SINGLE BLOCK mode and step through

the program until the thread start position is reached

Switch from AUTO to MANUAL mode - spindle stops and the threading tool is in clearance area

Select XZ screen display (absolute mode)

Switch to HANDLE mode for the Z-axis

While watching the XZ position display, move the handle in the same direction as the thread, until the tool reaches programmed Z-value, or it cannot move any further, whichever comes first

If the tool reached target Z-position first, tool setup is safe for threading

If the tool just about touched the part, but has not yet reached the target Z end position of the thread, tool setup needs adjustment by the difference between the programmed position and the actual position, plus some additional clearance

There are other testing methods available, for example, to use temporarily G01 *linear motion command*, instead of G32 *threading command*, without a part mounted in the spindle, of course.

In the non-threading mode, feed rate overrides are effective, whereas in threading mode, they are not!

By reading the current tool position on the screen display and comparing it with the programmed position, it will be possible to know whether a collision will happen or not. During the test, feed rate can be slowed down or stopped anytime. Main purpose of the program test is to establish safe working conditions *before* the threading takes place.

N. OTHER THREAD FORMS

Although the standard V-shape thread with the 60° included tip angle is the most common thread form for both metric and imperial threads, it is by no means the only form. There are many thread forms and shapes programmers encounter in machine shops, too numerous to list.

As an example of a different threading form, look at an ACME thread as a subject for discussion. In metric, there is an equivalent thread, called the *Metric Trapezoidal* thread. From programming perspective, both threads are almost identical. ACME thread has a 29° included thread angle, metric trapezoidal thread has a 30 angle and somewhat different geometry definition. Different inserts are used.

Main application of a trapezoidal type of thread is to *transmit a motion*, usually with a disengaging half-nut. Certain types of lead screws for conventional lathes use this type of thread as well. Programming trapezoid threads often requires a steady rest, since these threads may be quite long. An important consideration is the lead error accumulated over a long distance, discussed earlier - see *page 360*.

Thread Depth - ACME example

Every thread has its formulas and mathematical relationships. There are two basic formulas relating to an ACME thread depth. One is for threads of 10 TPI and *coarser*, the other for threads of 12 TPI and *finer*. For ACME threads 10 TPI and coarser, the thread depth formula is:

T_d	$0.500 P$	0.010

For the ACME threads 12 TPI and finer, the thread depth formula is modified only slightly:

T_d	$0.500 P$	0.005

+ where ...

$T_d = \text{Thread depth } P = \text{Thread pitch}$

Other threads in the trapezoidal group are *Stub ACME* or a *60 Stub ACME*. Programming trapezoidal threads is no more difficult than programming any V-shape thread, providing thread formulas and geometric details of the thread design are known to the CNC programmer.

There are other threads that can be encountered outside of the 60 category - *Whitworth* threads, *Square threads*, *API* threads (used in the petroleum industry), *Buttress*, *Aero*, *Dardelet* self-locking threads, *Round* threads, *Lebus* threads (require special control features), and a number of others. Thread and threading data can be found in specialized tooling catalogues and technical publications.

O. TAPERED THREAD

Programming procedure for a tapered thread is not significantly different than that for a straight thread. Tapered threading motion is along *two axes* simultaneously, rather than a single axis. The four basic motion steps are, therefore, almost identical to those for a straight thread:

Motion 1	Rapid from the start position to the thread diameter
Motion 2	Cut the thread (cutting along two axes)
Motion 3	Retract from thread
Motion 4	Return to the start position

When compared to a straight thread, the only programming differences for a tapered thread are in the first two motions - *Motion 3* and *Motion 4* remain unchanged. In the *Motion 1*, the starting tool position is determined by the physical orientation of the threading tool - whether it is used for an external or an internal thread.

For *external* thread forms, the starting position of the threading tool must always be *above the largest diameter* of the thread. For *internal* thread forms, the starting position must always be *below the smallest diameter* of the thread. This is the same requirement as for a straight thread, but for a tapered thread it takes on an additional importance. For examples of a tapered thread, evaluate the simplified drawing in *Figure 38-32*.

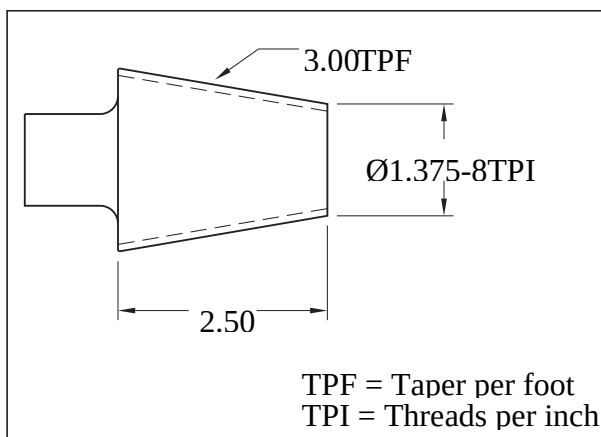


Figure 38-32

Basic drawing for a tapered thread example - program O3809

The thread is defined by its overall length (2.500), by its *front* diameter part (1.375), by its taper angle (3.0 inches taper per foot) and by its pitch (8 TPI). It is a single start thread and the program zero is at the front face of finished part. All pre-machining operations have been done for this example. The first programming consideration for this type of machining should always be the *thread depth*.

Depth and Clearances

From the previously established formula (p. 354), the external depth D of thread used in the program will be:

$$D = 0.61343 / 8 = 0.0766788 = 0.0767$$

Thread depth is measured axially and is not related to the thread angle (taper). Once the depth is established, clearances can be set - one in the front and one at the end of thread. Z-axis clearance amount will depend on the tool acceleration speed. Since threading feed rate will be programmed as F0.125, four times the lead rule of thumb would require a clearance to be 0.500 of an inch. Only 0.400 of an inch is sufficient with a relatively slow spindle speed of 450 r/min. End clearance can be smaller - there is enough open space at the thread end, and 0.200 is a reasonable clearance at this position, although a smaller clearance could be used as well.

For a tapered thread, always consider the *total* length of tool travel along each axis, *not* the actual thread length as per drawing - this is no different than for a single axis thread. Tool travel length in the example will be the combination of two selected clearances plus given thread length (along the Z axis):

$$0.400 + 2.500 + 0.200 = 3.100$$

The next step may not be always necessary, depending on actual method of programming. If a block-by-block G32 threading method is used, *both* start *and* end thread diameters will be needed for each pass. If a cycle G92 or G76 is used - the single distance between start and end diameter of a tapered thread is needed. This distance is programmed as a calculated dimension I of the one-block method or as a dimension R of the two-block threading cycle (second block). Previous examples were straight threads and I ® values were zero ($I0$ or $R0$). $I0$ or $R0$ can be omitted.

Taper Calculation

A thread taper has to be *calculated* to establish its start and end diameters. Any calculation method depends on how the taper is defined and dimensioned in the drawing. No part drawing will show the dimensions required for programming - they have to be calculated as part of the programming process, using one of two common methods.

One method uses the thread length and angle and can be calculated by applying standard trigonometric functions. The other method defines taper as a *ratio of its sides*. This method is often confusing to an inexperienced programmer. Typical ratios are defined in the part drawing directly, for example as 1:12, 1:16, etc., or indirectly, for example as the amount of *taper per foot* or, sometimes, as *taper per inch*. Keep one rule in mind:

Size of a taper is always measured on diameter

Standard North American pipe thread is a good example of a tapered thread. It is defined by a taper ratio of 1:16, which is equivalent to a $\frac{3}{4}$ of an inch per foot taper, measured on diameter and perpendicular to an axis. Pipe thread may also be defined with a given angle per side - *one degree, forty seven minutes, twenty three seconds (plus some leftover)*, or *1.789910608 decimal degrees*. For CNC programming, decimal degrees are preferred to degrees-minutes-seconds method (DMS or D-M-S), and many CAD drawings already reflect this preference. To understand all principles of a taper defined as the ratio of its sides, a definition followed by an example should help.

RATIO indicates the relationship between two values, expressed as a fraction

Both ratio values must be expressed in the same units and should be used in their lowest form of application ($\frac{1}{4}$ instead of $\frac{2}{8}$ or $\frac{4}{16}$). For example, the ratio of *3 units* to *4 units* may have these forms:

$$3 : 4 = 3 / 4$$

In terms of a taper definition, it means that for every 3 units change along one axis, there will be 4 units change along the other axis.

TAPER PER FOOT indicates the difference between two diameters over the length of one foot or 12 inches

The drawing example of a 3 inch taper per foot is equivalent to a 1:4 taper ratio, because

$$3 / 12 = 1 / 4 = 1 : 4$$

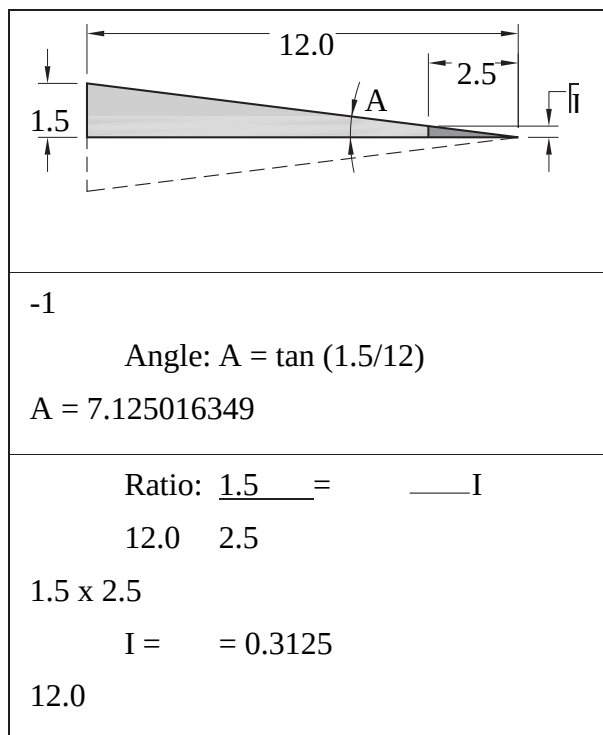


Figure 38-33

Taper thread calculations based on the example - clearances excluded

In CNC programming, we are only interested in calculating part diameters at the *beginning* and at the *end* of thread. These calculations can be done either by means of trigonometric functions, using the law of similar triangles, or by means of ratio calculations. Mathematical definitions relating to tapers start on *page 501*. Figure 38-33 shows angular calculation as well as a ratio calculation for the example.

Based on these principles, all required dimensions can now be calculated. Note that only *per side* or *radial* dimensions were used. In most programming applications, only one of the available methods will be required - either the angle or the ratio. There will always be an option to use the other method to verify accuracy of the calculations.

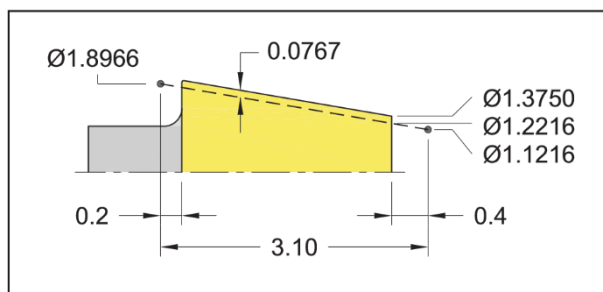


Figure 38-34

Calculated dimensions for taper thread example O3809

In Figure 38-34, the start and end diameters have been calculated using the angle and/or the ratio of sides method. Which results of the calculations will actually be used in the program will depend on the type of selected programming technique, such as using a block-by-block approach versus a cycle method. The details depend on the thread specifications and machine and control

features.

Block by Block Taper Thread

In block-by-block threading, taper thread programming is just as simple as programming a straight thread. To simplify the example, a straight infeed and nine threading passes will be used, for the total depth of 0.0767. The following nine depths must be applied at *both* ends of the thread. The first column lists depth of thread per pass, the second column lists front thread diameter, the third column is the end diameter. Front diameter is calculated at absolute Z-coordinate of Z0.4, the end diameter at Z-2.7:

Pass Depth			Front		End							

1	0.0165	1.2420	2.0170	2	0.0145	1.2130	1.9880	3	0.0120	1.1890	1.9640	4
0.0100	1.1690	1.9440	5	0.0080	1.1530	1.9280	6	0.0060	1.1410	1.9160	7	
0.0040	1.1330	1.9080	8	0.0030	1.1270	1.9020						
9	0.0027	1.1216	1.8966									

All requirements are available to write program O3808:

O3809

...

(G32 - TAPERED THREAD)

N46 T0500 M42

N47 G97 S450 M03

N48 G00 X2.5 Z0.4 T0505 M08

N49 X1.242 (PASS

N50 G32 X2.017 Z-2.7 F0.125

N51 G00 X2.5

N52 Z0.4

N53 X1.213 (PASS

N54 G32 X1.988 Z-2.7

N55 G00 X2.5

N56 Z0.4

N57 X1.189 (PASS

N58 G32 X1.964 Z-2.7

N59 G00 X2.5

N60 Z0.4

N61 X1.169 (PASS

N62 G32 X1.944 Z-2.7

N63 G00 X2.5

N64 Z0.4

N65 X1.153 (PASS

N66 G32 X1.928 Z-2.7

```

N67 G00 X2.5
N68 Z0.4
N69 X1.141                (PASS
N70 G32 X1.916 Z-2.7
N71 G00 X2.5
N72 Z0.4
N73 X1.133                (PASS
N74 G32 X1.908 Z-2.7
N75 G00 X2.5
N76 Z0.4
N77 X1.127                (PASS
N78 G32 X1.902 Z-2.7
N79 G00 X2.5
N80 Z0.4
N81 X1.1216               (PASS
N82 G32 X1.8966 Z-2.7
N83 G00 X2.5
N84 Z0.4
N85 G00 X12.0 Z4.5 T0500 M09
N86 M30
%
```

In the example, a radial infeed (plunge infeed) and pullout is used for clarity. The program will not change much if a compound infeed is used and/or an angular pullout from the thread is applied. Of course, more calculations will be needed for the new Z-start position (shifted from Z0.4).

Tapered Thread Using a Simple Cycle

In G92 threading cycle, thread taper is programmed as the radius *I* value, with specified direction *from* the end diameter *to* the start diameter:

```
G92 X.. Z.. I.. F..
```

The X-address in the cycle represents current thread diameter at the *end of cut*, Z-address is the thread end position, *I* is the difference *per side* between thread diameter at the *end* and thread diameter at the *start*. The *I* value must include an algebraic sign (only minus sign must be written), specifying the *direction* of taper inclination, in this case a negative value. Program O3810 will cut a tapered thread using G92 threading cycle.

O3810

...

(G92 - TAPERED THREAD)

N46 T0500 M42

N47 G97 S450 M03

N48 G00 X2.5 Z0.4 T0505 M08

```

N49 G92 X2.017 I-0.3875 Z-2.7(PASS
.125
N50 X1.988 (PASS2)
N51 X1.964 (PASS3)
N52 X1.944 (PASS4)
N53 X1.928 (PASS5)
N54 X1.916 (PASS6)
N55 X1.908 (PASS7)
N56 X1.902 (PASS8)
N57 X1.8966 (PASS9)
N58 G00 X12.0 Z4.5 T0500 M09
N59 M30
%
```

Note that the *I* distance of taper inclination is the difference between the *end* diameter of 1.8966 and the *start* diameter of 1.1216, divided by 2. The result is:

$$(1.8966 - 1.1216) / 2 = 0.3875$$

This *I* value (0.3875) needs a directional sign, to indicate the taper direction (its inclination from end point). The *I* value in the example is negative because the taper start diameter is *below* the thread end diameter, as viewed on a typical rear lathe. In the program, the *I* entry will be I-0.3875.

Tapered Thread and G76 Cycle

Both one-block and two-block multiple repetitive threading cycle G76 requires the taper inclination amount *not to be a zero*, if a tapered thread is machined. The *I* (one block method) or *R* (two block method - second line) value in the cycle specifies the difference *per side*, so called *radial distance*. It can be a positive or a negative value, indicating the direction between end and start diameters of the taper.

Remember that the X-diameter is always programmed at the *end* of thread and the *I* amount supplies the actual taper height as well as its inclination (taper ratio per side). On CNC lathes with the X+ axis direction upwards from the center line (rear lathes), an increasing taper diameter will require a negative *I* amount, and a decreasing taper will require a positive *I* amount. The programmed *I* amount is always a *single* value, measured on radius, not diameter *Figure 38-35* illustrates the concept for rear CNC lathes.

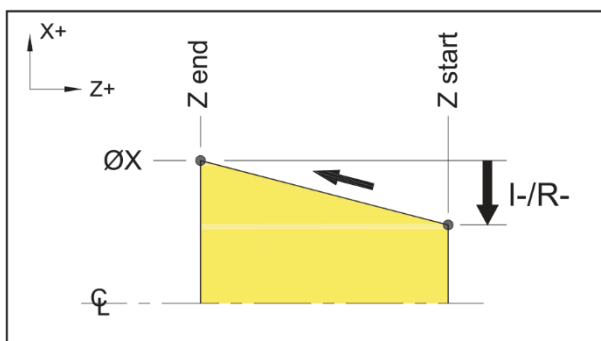


Figure 38-35

Inclination direction of an EXTERNAL tapered thread in G76 cycle

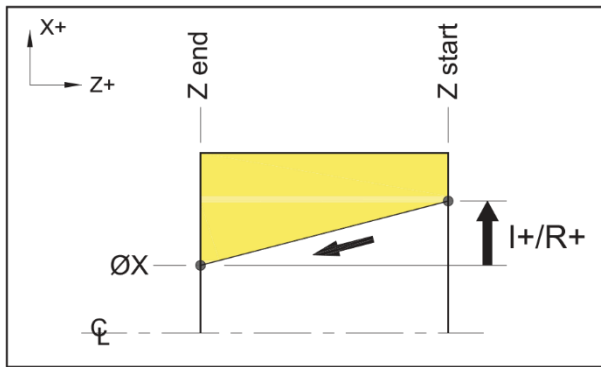


Figure 38-36

Inclination direction of an INTERNAL tapered thread in G76 cycle

Basic G76 cycle will be maintained but the *I* or *R* address will be added - a non-zero value must be programmed:

O3811 (G76 ONE BLOCK - TAPERED THREAD) ...

N46 T0500 M42

N47 G97 S450 M03

N48 G00 X2.5 Z0.4 T0505 M08

N49 G76 X1.8966 Z-2.7 I-0.3875 K0.0767 D0140

F0.125

N50 G00 X12.0 Z4.5 T0500 M09

N51 M30

%

O3812 (G76 TWO BLOCKS - TAPERED THREAD) ...

N46 T0500 M42

N47 G97 S450 M03

N48 G00 X2.5 Z0.4 T0505 M08

N49 G76 P011060 Q004 R0.002

N50 G76 X1.8966 Z-2.7 R-0.3875 P0767 Q0140

F0.125

N51 G00 X12.0 Z4.5 T0500 M09

N52 M30

%

If this method can be used for threading, G76 cycle is the best choice. It offers the fastest program generation as well as the best opportunities for on-machine editing.

P. MULTISTART THREAD

Most threads have only one start, suitable for most applications. The most common purpose of a multi-start thread is to transfer a *precision* motion very rapidly over a relatively long distance. Note the word *precision* - a coarse thread can be used to transfer motion rapidly, but with very

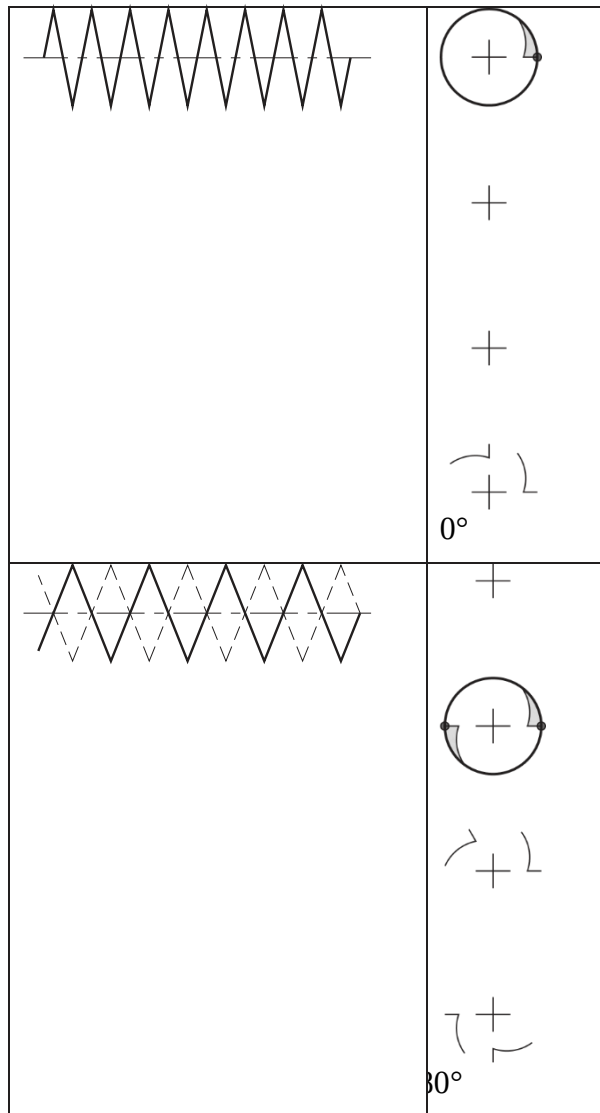
little precision. Fine thread has the precision but transfers any motion slowly. An example of precision multi-start threads are internal designs of some cameras zoom lenses.

For programmers, there are some unique considerations for multi-start threads. It is important that the start position for each thread is in such a location, that when viewed from the thread end, each start on the circumference will be divided in equal angular increments. Also important is to maintain equal thread profile when viewed from the thread cross section. To achieve these conditions, two programming tools are available:

Controlled thread start position

Controlled feed rate

Figure 38-37 shows symbolically views of the thread cross sections and end views.



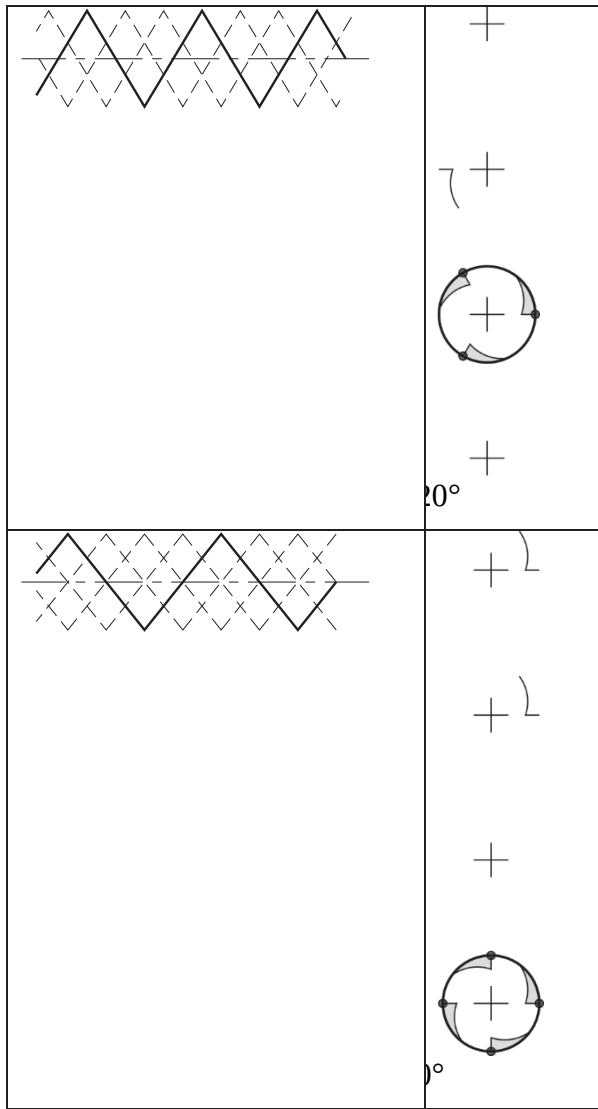
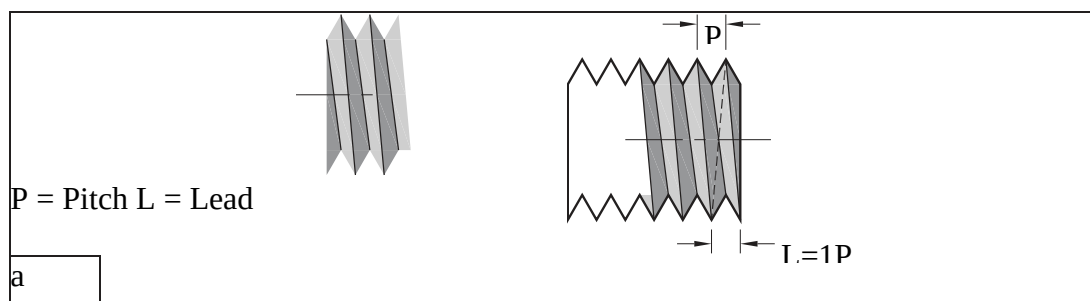


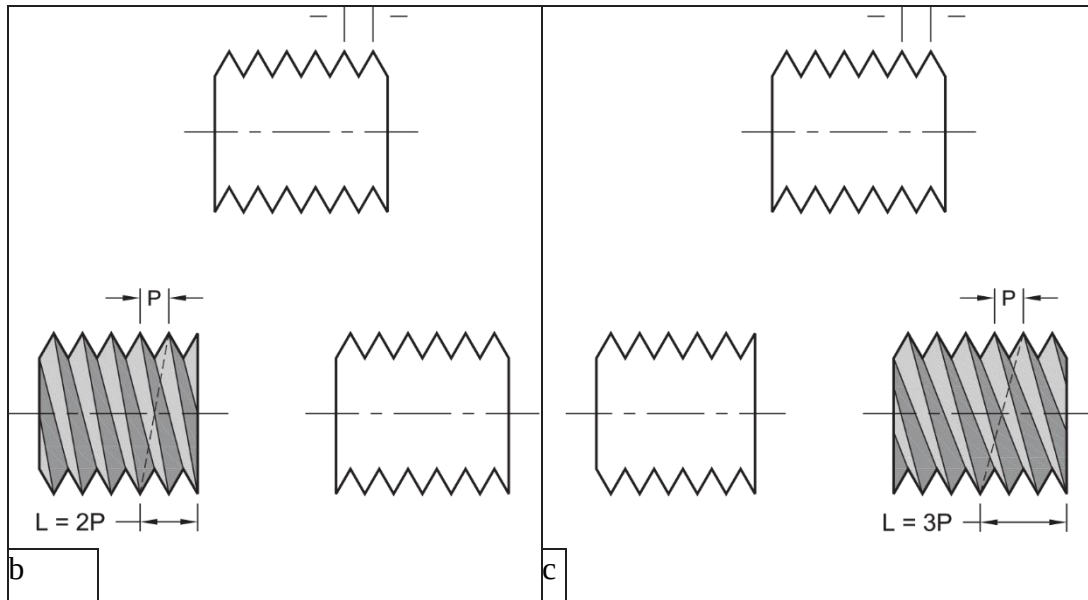
Figure 38-37

Schematic representation of 1,2,3 and 4 start threads (dot = start)

In the illustration are four examples of the cross sections (left) and the end views (right) of a single start thread (top), double start (one below), triple start (two below) and a quadruple start (three below).

Even when the examples are represented only symbolically, specified thread *pitch* is maintained in all instances. Also note the *equal* distribution of each thread start.





Equal distribution is the key for successful multi-start threading, when the threaded part is viewed along its center line. Correct spacing is achieved by a programmed *shift amount* from one thread start to the next.

Threading Feed rate Calculation

Threading feed rate is always the *lead* of thread, never the *pitch*. For a single start thread, lead and pitch have the same value - for a multi-start thread, they do not. For example, take a single start thread of 16 TPI. Here, the lead *and* pitch are both 0.0625, so the feed rate is F0.0625. If the drawing specifies the thread as 16 TPI, but indicates a *double start*, (for example 3.0-16 TPI 2-START), that means the *pitch* of thread will remain unchanged (0.0625), but its *lead* will double to 0.1250. Therefore, programmed feed rate for double start thread with pitch of 0.0625 will be F0.125. The pitch multiplication will always depend on the *number of threads starts*. That means a triple start thread will have the feed rate three times the pitch, quadruple start thread four times, and so on.

Figure 38-39

Multi-start thread - relationship between pitch and lead:

(a) Single start thread	Lead = Pitch = $1P = P$
(b) Double start thread	Lead = $2P$
(c) Triple start thread	Lead = $3P$

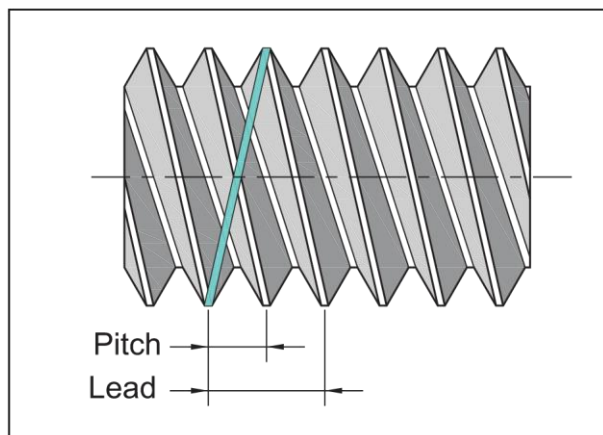


Figure 38-38

PITCH and LEAD relationship for a two start thread

In Figure 38-38, the relationship of pitch and lead of a double start thread is shown. The same logic that applies to a double start thread, also applies to triple, quadruple, etc., threads. Feed rate calculation is identical for *all* threads:

	Number of starts
Feed rate	_____
	TPI

Figure 38-39 shows the relationships of pitch and lead for some more common multi-start threads - the same pitch-lead relationship is maintained proportionately. Metric threads use only pitch, so the above formula can be changed to use pitch instead of threads per inch (metric *and* imperial):

Feed rate	Number of starts Pitch

Shift Amount

Feed rate is not the only consideration for programming a thread with two or more starts. The other - equally important factor - is the programmed amount of *tool point shift*. This shift guarantees that each start will be in synchronized relationship to all other starts. When one thread is finished, the tool start position has to be *shifted* in Z-axis only and always by the *pitch* distance. Formula for the tool shift amount is always equal to the pitch:

Shift amount Pitch

Shift has to be programmed *for each start after* the first one - the number of shifts is *one less than number of starts*:

Number of shifts = Number of starts - 1

Note - this formula is valid even for a single start thread, where no shift is required (1 start - 1 = 0 shifts).

A few methods can determine *when* the tool shift is to be programmed. The first method, for a double start thread, is to program one thread to its full depth, then shift out and cut the second thread to its full depth. Another method, for the same thread, is to cut one pass of the first thread, shift *out*, cut the same pass for the second thread, shift *in*, cut the second pass for the first thread, shift *out* again and repeat the process until both threads are completed to the full depth. This approach applies to any number of starts.

Obvious advantage to the first method is programming ease. On negative side, if the tool cutting edge wears out on the first thread, second thread will not be as accurate. The advantage of the second method is that tool wear will be equally distributed over both threads, although programming will require a lot more effort, which presents the negative side. Additional problem is that in many hard materials, the thread edge life may suffer from extensive material removal.

Application Example

To illustrate a sample multi-start thread application, previous thread specifications are based on program O3804:

Number of threads per inch is twelve (12 TPI)

Number of starts is TWO (double start thread)

Thread is cut as external at 3.000 nominal diameter

Calculated thread depth is 0.0511 (0.61343 / 12)

Although the block-by-block programming method G32 can be used for special applications, acceptable results can be achieved in many threading applications by using the G92 or G76 cycles,

with less programming, as well as the gain of easier editing at the machine. G76 two-block version is shown here, as it covers most modern controls. Although two versions will be presented, neither one may produce the best results in certain materials - cycles are always bit of a compromise between power and programming convenience.

The first program, O3813, shows the first thread fully completed *before* the second thread starts (cutting feed rate is F0.1667, *not* F0.0833):

O3813 (G76 - EACH START TO FULL DEPTH) ...

N59 T0500 M42

N60 G97 S450 M03

N61 G00 X3.3 Z0.5 T0505 M08 (START POSITION)

N62 G76 P011060 Q004 R0.002

N63 G76 X2.8978 Z-1.6 R0 P0511 Q0210 F0.1667

N64 G00 X3.3 Z0.5833 (SHIFTED START POSITION)

N65 G76 P011060 Q004 R0.002

N66 G76 X2.8978 Z-1.6 R0 P0511 Q0210

N67 G00 X12.0 Z4.5 T0500 M09

N68 M30

%

Three major features of the above program should be mentioned. First, the original start point has been increased from Z0.25 to Z0.5, to allow feed rate acceleration in the air (block N61). Second, in blocks N62 and N63, the G76 cycle is used as in program O3804, with a major exception the modal feed rate is *doubled* for a *double start* thread. Third, added block N64 - this block changes the start position for the *second* thread start. As the original start position was Z0.5, the new position must be increased by the pitch to Z0.5833. W0.0833 can be used instead of Z.

Quality of Multi-start Threads

One of the potential problems using G76 cycle for multiple threads is the *tool wear*. For soft materials, short threads and a few starts, the previous example is quite suitable to produce quality threads. Long multi-start threads, threads with many starts, threads machined in hard materials, etc., present a situation where the threading tool may significantly wear out *before* all threads are machined to their full depth. G76 cycle itself does not provide for a situation such as this one, but a resourceful programmer should be able to find a suitable workaround.

The easiest method is to use two identical tools - one for thread *roughing* and one for thread *finishing*. In program O3814, the main difference is the *shortened cutting depth*.

O3814 (G76 - ROUGH AND FINISH - TWO TOOLS) ...

N59 T0500 M42 (T05 = ROUGHING TOOL)

N60 G97 S450 M03

N61 G00 X3.3 Z0.5 T0505 M08 (START POSITION)

N62 G76 P011060 Q004 R0.002

N63 G76 X2.9098 Z-1.6 R0 P0451 Q0210 F0.1667

N64 G00 X3.3 Z0.5833 (SHIFTED START POSITION)

N65 G76 P011060 Q004 R0.002

~~N66 G76 X2.9098 Z-1.6 R0 P0451 Q0210~~
 N67 G00 X12.0 Z4.5 T0500 M09
 N68 M01
 N69 T0700 M42 (T07 = FINISHING TOOL)
 N70 G97 S450 M03
 N71 G00 X3.3 Z0.5 T0707 M08 (START POSITION)
 N72 G76 P011060 Q004 R0.002
 N73 G76 X2.8978 Z-1.6 R0 P0511 Q0450 F0.1667
 N74 G00 X3.3 Z0.5833 (SHIFTED START POSITION)
 N75 G76 P011060 Q004 R0.002—
 N76 G76 X2.8978 Z-1.6 R0 P0511 Q0450
 N77 G00 X12.0 Z4.5 T0700 M09
 N78 M30
 %

Roughing tool left 0.006 stock per side for finishing that is reflected in blocks N63 and N66 (underlined). Finishing tool uses the same program as before, but its Q-depth was

increased to minimize air-cutting - most of the material has been removed with the roughing tool. ***Note that all other data are identical for both tools!***

Although this method is the easiest to program, it requires much more effort during setup at the machine. Both tools have to be the same type, they have to be set very accurately, so they reach each start thread at the same location. Also, geometry and wear offsets have to be set correctly for both tools and maintained properly - if one wear offset for one tool is changed, it may often be necessary to change the other as well.

When applying this technique, tool wear will be evenly distributed between all threads and should result in a longer tool life and high quality multi-start threads.

As an alternate method that also incorporates roughing and finishing cuts is a program that uses only *a single* tool, rather than two tools. This method is shown in the next program - O3815:

O3815 (G76 - ROUGH AND FINISH - ONE TOOL) ...

N59 T0500 M42
 N60 G97 S450 M03
 N61 G00 X3.3 Z0.5 T0505 M08 (START POSITION)
 N62 G76 P011060 Q004 R0.002
 N63 G76 X2.9098 Z-1.6 R0 P0451 Q0210 F0.1667
 N64 G00 X3.3 Z0.5833 (SHIFTED START POSITION)
 N65 G76 P011060 Q004 R0.002
 N66 ~~G76 X2.9098 Z-1.6 R0 P0451 Q0210~~
 N67 G00 X3.3 Z0.5 (START POSITION)
 N68 G76 P011060 Q004 R0.002
 N69 G76 X2.8978 Z-1.6 R0 P0511 Q0450 F0.1667

N70 G00 X3.3 Z0.5833 (SHIFTED START POSITION)

N71 G76 P011060 Q004 R0.002

N72 G76 X2.8978 Z-1.6 R0 P0511 Q0450

N73 G00 X12.0 Z4.5 T0500 M09

N74 M30

%

Although this method makes setup much more efficient, it is not recommended for all multi-start threading applications, as it does not fully solve issues associated with tool wear. For short job runs, it may work quite well and be a reasonable compromise.

This section has presented some special options that can be useful in certain thread cutting situations. Interpret all presented ideas as - *just ideas*. Threading jobs vary greatly and there is not a single solution to all threading problems.

Q. THREAD RECUTTING

When thread cutting is done, it should be checked for quality *before* the part is removed from the machine. Once the part is removed, any subsequent re-clamping will need a great effort in order to recut the thread. While the part is clamped, the first threading pass will start at a random place of the cylinder circumference. Each subsequent pass will be automatically synchronized to start at the *same* position. As long as the threaded part remains clamped, this synchronization is assured.

Before Part Removal

Prevention is always worth the extra effort necessary. There is one or two methods available to prevent thread recutting - just follow one important rule:

Do not unclamp part until the thread is checked

When threading operation is completed, check the thread *at the machine*, while the part is still clamped. This inspection can be done simply enough, if suitable thread gages are available. If a matching part is available, it may provide another checking method, although not one recommended by quality inspectors. Otherwise, you may have to go as far as using the rather awkward but reliable three-wire method.

After Part Removal

Once the threaded part is removed from the holding device, its thread should be correct. In case it is not - and in spite of all prevention methods - what can be done? This subject is aimed mainly towards CNC lathe operators, but programmers can learn from it as well.

First, make sure to program a tool wear offset. Second, program M00 function at the end of each threading operation, before any other machining, even for the last tool. If the thread has to be recut after removal - while the part is still clamped, the CNC operator has to follow several steps:

Re-clamp the threaded part to run concentric w/spindle

Set the X-axis offset large enough, so the threading tool moves above the thread (external threading) or below the thread (internal threading)

Visually align the threading tool tip with the thread already completed (only as accurate as one's eye)

31. SUBPROGRAMS

The length of a CNC program is usually measured in the number of characters such program contains. This number is similar to the number of bytes, if the program is stored on a computer disk. Physical length of a program is usually not an issue for most jobs. Program length will vary, depending on the complexity of work, number of tools used, method of programming and other factors. Generally, the shorter the program, the less time is needed to write it, and the less space it will occupy in CNC memory. Short programs also reduce the possibility of a human error, because they are easily checked, modified and optimized. Virtually all CNC systems offer features designed to shorten the length of a program to some extent and make the programming process easier, more efficient and less prone to errors. Typical examples of this type of programming are fixed cycles, multiple repetitive cycles and custom macros. This chapter describes the structure, development and applications of another method of efficient program preparation the use of *subprograms*.

A. MAIN PROGRAM AND SUBPROGRAMS

A CNC program is a series of instructions, assigned to different tools and operations. If such a program includes two or more *repetitive* instructions, its structure should be changed from a single long program to two or more separate programs. Each repetitive instruction is written only once and called when required. This is the main concept of subprograms. *Figure 39-1* shows a typical part layout repeated at different locations.

Each program must have its own program number and is stored in the control memory. Programmer uses special M-function to call one program from another. The *first* program that calls another program is called the *main program*; all other programs are called *subprograms*. Main program is never called by a subprogram - it becomes the top level of all programs. Subprograms can also be called from other subprograms, up to a certain number of nesting levels. When a program containing subprograms is used, always select the main program, never the subprogram. The only time a subprogram is selected at the control is for editing purposes. In some reference materials, subprograms are also called *subroutines* or *macros*, but the term *subprogram* is used most often, and the word *macro* could have a different meaning altogether.

Subprogram Benefits

Any frequently programmed order of instructions or unchanging block sequences can benefit from becoming a subprogram. Typical applications for subprogram applications in CNC programming are:

Repetitive machining motions

Functions relating to tool change

Hole patterns

Grooves and threads

Machine warm-up routines

Pallet changing

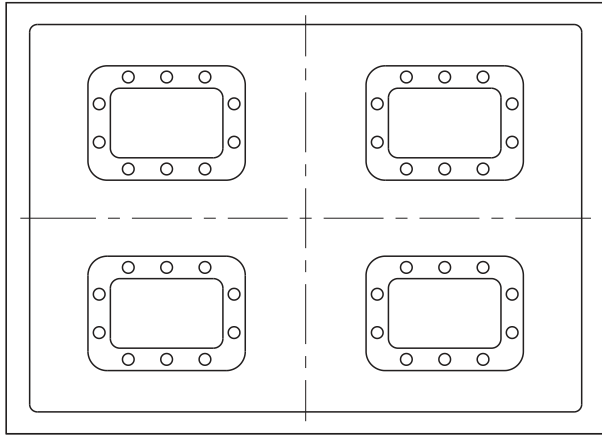


Figure 39-1

Example of a part requirement suitable to be used as a subprogram

Special functions ... and others

Structurally, subprograms are similar to standard programs. They use the same syntax rules and look and feel the same. Often, it may not be easy to see the difference between a regular program and a subprogram at a casual glance. A subprogram can use absolute or incremental data input, as necessary. Subprograms are loaded into the CNC system memory just like other programs. When properly implemented, they offer several benefits:

Program length reduction

Program error reduction

Programming time and effort reduction

Quick and easy modifications

Not every subprogram will provide all benefits, but even one benefit should be a good reason to use subprograms.

Identification of Subprograms

The first step towards a successful application of subprograms is the *identification and isolation of repetitive* programming sequences. For example, the next six program blocks represent a machine zero return for a typical horizontal machining center, at the start of program:

N1 G20

N2 G17 G40 G80 (STATUS BLOCK)

N3 G91 G28 Z0 (Z-AXIS RETURN)

**N4 G28 X0 Y0 (X AND Y AXES
RETURN)**

N5 G28 B0 (B-AXIS RETURN)

**N6 G90 N7 ... (ABSOLUTE
MODE)**

These blocks represent a typical sequence of commands that will be repeated *every time* a new program for that machine is written. Such a program may be written many times a week, each time repeating the same sequence of instructions. To eliminate any possibility of an error, frequently used order of blocks can be stored as a separate program and identified by a unique program number. Then, it can be called up at the top of any main program. This stored programming sequence will become a *subprogram* a branch, or an extension, of the main program.

B. SUBPROGRAM FUNCTIONS

A subprogram must be recognized by the control system as a *unique* type of program, *not as a main program*. This distinction is accomplished with two miscellaneous functions, normally applicable to subprograms only:

M98	Subprogram call function
M99	Subprogram end function

Subprogram *call function* M98 must always be followed by the subprogram number P--. Subprogram *end function* M99 terminates the subprogram and transfers processing back to program it originated from (a main program or a subprogram). Although M99 is mostly used to end a subprogram, it may also be rarely used in the main program, replacing the M30 function. In this case, the program will run 'forever', or until the *Reset* key is pressed.

Subprogram Call Function

Function M98 calls up a previously stored subprogram from another program. If used only by itself in a block, it will result in an error. M98 is an incomplete function - it requires two *additional* addresses to become effective:

Address P identifies the selected subprogram number

Address L or K identifies the number of subprogram repetitions (L1 or K1 is the default)

For example, a typical subprogram call block includes the M98 function and the subprogram number:

N167 M98 P3951

In block N167, subprogram O3951 is called from CNC memory, to be repeated *once* - L1 (K1) counter is the default, depending on the control. Subprogram must be stored in the control before being called by another program.

M98 blocks that call subprograms may also include additional instructions, such as rapid tool motions, spindle speed, feed rate, cutter radius offset number, etc. On most controls, if included in the same block as subprogram call, the additional data will be passed to the contents of the subprogram. The following subprogram call block also contains a tool motion in two axes:

N460 G00 X28.373 Y13.4193 M98 P3951

This block executes the rapid motion first, *then* it calls the subprogram. The order of words in a block makes no difference to the block execution:

N460 M98 P3951 G00 X28.373 Y13.4193

results in the same machining order as if the tool motion preceded the subprogram call but looks illogical.

Subprogram End Function

When the main program and subprogram coexist in the control, they must differ by their program numbers. During processing, they will be treated as one continuous program, so a distinction must be made for the program end function as well. The end of program function is M30 or, less frequently, M02. Subprogram must be terminated by a different function. Fanuc uses M99 for that purpose:

O3951 (SUB-1) Subprogram start

... ..

M99 Subprogram end

%

When a subprogram terminates, control returns program processing to the program of origin - it will *not* terminate the main program - that is the exclusive function of M30. Additional parameters may also be added to the M99 subprogram end, for example a block skip code [/], a block number to return to upon exit, etc. Note that the stop code symbol (the % sign) is used in the same manner for a subprogram, as for any main program. Subprogram termination is important and must always be done right. It sends two very important instructions to the control system:

To terminate the subprogram

To return to the block following subprogram call

Never use the program end function M30 (M02) to terminate a subprogram - it will *immediately cancel all program processing* and reset the control. Program end function does not allow program execution of any blocks beyond the block that contains it.

Normally, subprogram end M99 returns the processing to the block *immediately following* subprogram call M98. This concept is illustrated in *Figure 39-2* (without block numbers) and described next.

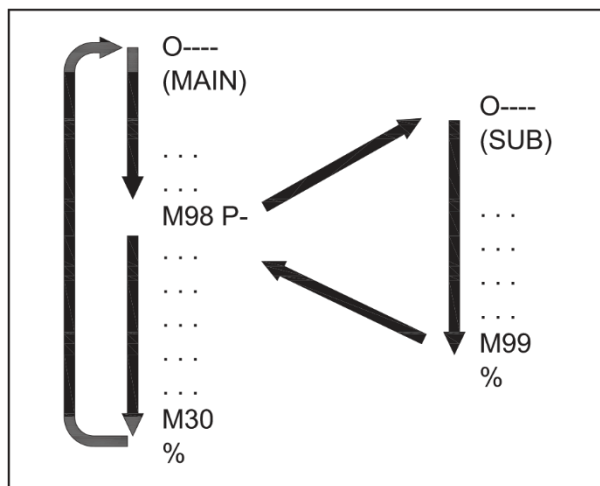


Figure 39-2

Flow of a program processing with a single subprogram

Block Number to Return to

In most programs, M99 function is programmed by itself, as a standalone entry, and as the *last* instruction in the subprogram. Usually, there are no other commands included with it in the same block. M99 function causes the subprogram to terminate and transfers its execution to the *next block* of the program it originated from. For example,

N67 M98 P3952 (SUBPROGRAM CALL) N68 ... (BLOCK TO RETURN TO FROM O3952)

N69 ...

N70 ...

executes block N67 by calling subprogram O3952. When subprogram O3952 is processed, control returns to the original program and continues processing instructions from the block N68, which is the *block to return to*.

Special Applications

For some special applications, such as in bar feeding, it may be necessary to specify a *different* block number to return to, rather than using the *next block* default. If the programmer finds this option useful for some jobs and uses this technique, the P-address must be included with M99:

M99 P..

In this format, the P-address represents the block number to return to - *from* the completed subprogram. This block number must be present in the program of origin. For example, if the main program contains these blocks,

(MAIN - PROGRAM) ...

N67 M98 P3952

N68 ...

N69 ...

N70 ...

and the subprogram O3952 is terminated by

O3952 (SUB) ...

...

M99 P70

%

the calling program processing will continue from block N70 (the main program in the example), *bypassing* blocks N68 and N69. This kind of application is not very common it requires suitable type of work, in addition to the thorough understanding of sub-programming principles.

Address P has a different meaning when used with M98 and M99 functions

Daily applications of this powerful programming method are not common, but the feature is an item to be explored by inquiring programmers. Various associated applications will include other programming tools, such as a combination with a block skip function, using the slash code /.

Number of Subprogram Repetitions

A very important subprogram call feature is the address L or K, depending on the control model. This address specifies the number of subprogram repetitions - *how many times* the subprogram has to be repeated before processing resumes in the original program. In most programs, a subprogram will be called only once, then the original program will continue.

Programs that require a *multiple* subprogram repetition before proceeding with the rest of the original program are common. For comparison, a single use of the subprogram O3952 could be called up from the program of origin as:

N167 M98 P3952 L1 (K1)

This is a correct program block, but the L1/K1 counter does not have to be programmed at all. It can be safely ignored - control unit always defaults to only *one* repetition.

If no address L/K is specified, the default value is always L1/K1

N167 M98 P3952 L1 (K1) is identical to **N167 M98 P3952** *Note - In the following examples, substitute K for every L listed, if required by the control system*

Number of repetitions for some control models range between L0 and L9999 and the L address other than L1 must always be programmed. Some programmers write the full block, even for a single repetition, rather than counting on the default conditions of the control system. The choice is a personal preference.

Repetition Count Variation

Some Fanuc controls *do not* accept the L/K address as the number of repetitions and use a different format. On these controls, a single subprogram call is the same:

N342 M98 P3952

This block calls the subprogram only once, as no special request has been used. In order to repeat the subprogram, call four times, instead of programming

N342 M98 P3952 L4 (K4)

program the requested number of repeats *directly after* the P address, in a single statement:

N342 M98 P43952 is the same as N342 P00043952

The result is identical to the other version - subprogram will be repeated four times. *The first four digits are reserved for the number of repeats, the last four digits define the subprogram number.* For example,

M98 P3950 is the same as M98 00013950

assumes a single repetition of subprogram O3950. In order to repeat O0050 subprogram 39 times, program

M98 P390050 or M98 P00390050

The maximum number of repetitions does not change for the 0/16/18/20/21 controls - it is represented by the first four digits, to the maximum of 9999.

M98 P99993952

repeats the execution of subprogram O3952, nine thousand, nine hundred and ninety nine times, the maximum number of repetitions available (some old models may have the maximum of only 999 times).

L0/K0 in a Subprogram Call

There is no mystery in using the L/K counter greater than one to repeat a subprogram. This is a common application. Fanuc also offers a *zero* number of repetitions, in the form of L0/K0. When can the L0/K0 be programmed? Would anybody want to repeat a subprogram *zero times*?

There are some good reasons. Observe *Figure 39-3*. The five hole pattern has to be spot drilled, drilled and tapped.

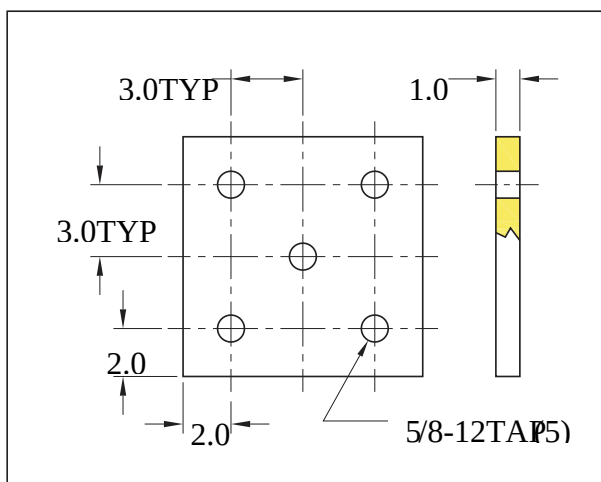


Figure 39-3

Sample drawing used for subprogram development Used in programs O3901, O3902 and O3953

For the spot drill (0.750), G82 cycle is used with 0.2 seconds dwell to Z-0.3275 depth. For tap drill, G81 cycle is used, and for tapping 5/8-12 tap, G84 cycle is used. Spot drill prepares the hole

for drilling and makes a 0.015 chamfer. Tap drill will be 35/64 drill (0.5469), to open up the hole for 5/8-12 tap:

O3901

(TOOL 1 - 90-DEG SPOT DRILL - 3/4 DIA)

N1 G20

N2 G17 G40 G80 T01

N3 M06

N4 G90 G00 G54 X2.0 Y2.0 S900 M03 T02

N5 G43 H01 Z1.0 M08

N6 G99 G82 R0.1 Z-0.3275 P200 F3.0 (LL HOLE)

N7 X8.0 (LR HOLE)

N8 Y8.0 (UR HOLE)

N9 X2.0 (UL HOLE)

N10 X5.0 Y5.0 (MIDDLE

N11 G80 Z1.0 M09 (HOLE)

N12 G28 Z1.0 M05

N13 M01

(TOOL 2 - 35/64 DRILL)

N14 T02

N15 M06

N16 G90 G00 G54 X2.0 Y2.0 S840 M03 T03 N17 G43 H02 Z1.0 M08

N18 G99 G81 R0.1 Z-1.214 F11.0

N19 X8.0

N20 Y8.0

N21 X2.0

N22 X5.0 Y5.0

N23 G80 Z1.0 M09

N24 G28 Z1.0 M05 N25 M01

(TOOL 3 - 5/8-12 TAP)

N26 T03

N27 M06

N28 G90 G00 G54 X2.0 Y2.0 S500 M03 T01

N29 G43 H03 Z1.0 M08

N30 G99 G84 R0.4 Z-1.4 F41.0

N31 X8.0

N32 Y8.0

N33 X2.0

N34 X5.0 Y5.0
N35 G80 Z1.0 M09
N36 G28 Z1.0 M05
N37 G28 X5.0 Y5.0
N38 M30
%

This type of program uses repeating XY coordinates for each tool (spot drilling, drilling, tapping). In order to make this program more effective, all *repeating* program blocks will be collected into a subprogram and used much more efficiently. Here is a pattern of holes separated from the long program that also includes G80Z1.0M09, as the standard end of *any* active fixed cycle (G20 mode):

X2.0 Y2.0
X8.0
Y8.0
X2.0
X5.0 Y5.0
G80 Z1.0 M09

Only a small effort is needed to reformat the existing program and separate it into a main program and a subprogram that stores the repeating machining pattern. Isolated XY coordinates of all five holes in the pattern are included:

O3953 (SUBPROGRAM)
(FIVE HOLE PATTERN)
N1 X2.0 Y2.0
N2 X8.0
N3 Y8.0
N4 X2.0
N5 X5.0 Y5.0
N6 G80 Z1.0 M09
N7 M99
%

This subprogram can be called from the main program, in this example, from a new program O3902. Programming L0 prevents double cutting of the first hole:

O3902 (MAIN PROGRAM)
(TOOL 1 - 90-DEG SPOT DRILL - 3/4 DIA)
N1 G20
N2 G17 G40 G80 T01
N3 M06
N4 G90 G00 G54 X2.0 Y2.0 S900 M03 T02
N5 G43 H01 Z1.0 M08

N6 G99 G82 R0.1 Z-0.3275 P200 F3.0 L0
 N7 M98 P3953
 N8 G28 Z1.0 M05 N9 M01
 (TOOL 2 - 35/64 DRILL)
 N10 M06
 N11 T02
 N12 G90 G00 G54 X2.0 Y2.0 S840 M03 T03
 N13 G43 H02 Z1.0 M08
 N14 G99 G81 R0.1 Z-1.214 F11.0 L0
 N15 M98 P3953
 N16 G28 Z1.0 M05 N17 M01
 (TOOL 3 - 5/8-12 TAP)
 N18 M06
 N19 T03
 N20 G90 G00 G54 X2.0 Y2.0 S500 M03 T01
 N21 G43 H03 Z1.0 M08
 N22 G99 G84 R0.4 Z-1.4 F41.0 L0
 N23 M98 P3953
 N24 G28 Z1.0 M05
 N25 G28 X5.0 Y5.0
 N26 M30
 %

In the program, initial XY motion for each cutting tool will position the cutter at the *first* hole of the machining pattern. All fixed cycles used in the program will start at the first hole of the pattern. Since the first hole definition is included in the subprogram, *as well as in the main program*, programming L0 in the fixed cycle call is mandatory, else the first hole of the pattern will be machined twice. This is a classic application of L0/K0 as it relates to fixed cycles, but not to subprograms. Subprogram O3953 may also include standard machine zero return block G28Z1.0M05, as it repeats after each M98 call in the main program O3902. This practice is correct but *not* recommended, as it does not follow a clearly defined program structure.

C. SUBPROGRAM NUMBERING

To keep track of subprograms is much more important than keeping track of regular programs. Always make sure to know *exactly* what subprograms are available and how they are used, what is their purpose. A single subprogram may be used in many other programs and proper subprogram identification technique is extremely important.

Control unit directory of programs does not distinguish between program numbers and subprogram numbers. The control system recognizes a subprogram call only by its programmed format, the miscellaneous function M98, followed by the P.. subprogram number statement.

All this means that subprogram number is assigned at the *programming* level, not at the *machine operation* level. It is the programmer's responsibility, *not* the CNC operator's, to assign subprogram numbers. Programmer has a great flexibility in organizing all subprograms and their

identification - in fact, any programmer can design and set up certain basic rules and related standards. Many of the rules governing format of main programs also apply to subprograms. Remember these four main points:

If used at all, program number is commonly specified by the letter O, followed by four or five digits, depending on the control system

If at all, program number can be specified by the colon symbol, commonly: for the ISO format, followed by up to four or five digits, depending on the control system setting

Main program number - O or: - cannot be negative or equal to zero

Subprogram number cannot be negative or equal to zero

Within the allowed range, *any* number can be assigned to *any* main program or subprogram. Some programmers do not use program numbers at all. This approach is acceptable for most controls, but only if the application does not require subprograms. In most cases, main program numbers can be assigned by the machine operator. On the other hand, to maintain *control* of subprograms, program numbers become very important. The first step is to get organized. This is even more important if subprograms are designed to be called up by *many other programs at different times*. There is not one *best* method, but some proven suggestions offer an idea how to approach the subject of program numbering and develop a personal approach.

For example, in this digital book, all main programs are numbered consecutively, with the first two digits corresponding to the chapter number. In this chapter, the method also applies to subprograms, but the last two digits are arbitrarily increased by fifty, for example O3953 will be the third subprogram example in the chapter. Feel free to adapt this method to any reasonable format.

Organized Approach

Suggested programming approach is based on the understanding that CNC memory is *not* used as a storage media for all part programs ever made. Control system memory capacity is *always* limited. At one point, this limit will be reached and there will be no more space left to accommodate more programs. Good program organization is one that uses CNC system memory storage only for the *current program*, perhaps a few more that are to be used soon.

If a unique program number is assigned by the *machine tool operator* during setup, this situation needs some good management as well. On some controls, main program number on the written copy will not always load automatically, so it is not really required. That means if an arrangement is made with the shop supervisor that the CNC operator stores all main programs using only *three* digits for regular program numbers 1-999; then there will be four digit numbers 1000-9999 available for subprograms. This available range is more than enough for most manufacturing applications. Such an approach presents a good management over those subprograms whose numbers selected. All four-digit subprogram numbers can be documented, logged and subsequently called from any program, main or another subprogram, without a fear of duplication or a part program number mismatch.

Subprograms should always be documented in some logbook, complete with detailed descriptions, independently from all programs of origins. This way, such subprograms can be used when needed, often at a short notice, regardless of the program for which they have been originally written. Such a method allows to organize all subprograms by their series number (*i.e.*, 1000, 2000, 3000, etc., or 1100, 1200, 1300, etc.), for either the type of CNC machine, the type of subprogram, or the type of machining operation.

Individual subprograms have to have assigned program numbers that are unique. Program number assigned to a subprogram is called together with the M98 function and P address. Such a combination of the two words, M98 P., is the *minimum requirement* for a subprogram call from another program.

Using the example (early in this chapter) of the machine zero return sequence for a four axis vertical machining center, a subprogram can be created (with an assigned number O3954) for the blocks representing all needed commands units selection G20 or G21 is *not* included:

O3954 (MACHINE ZERO RETURN)

N101 G17 G40 G49 G80

N102 G91 G28 Z0

N103 G28 X0 Y0

N104 G28 B0

N105 G90

N106 M99

%

Units selection should always be used in the main program, for flexibility. Once the machine zero return subprogram has been designed and stored in the memory, every main program can start by calling subprogram O3954:

O3903 (MAIN PROGRAM)

ART ABC-123)

N1 G20 *Units used for this program*

N2 M98 P3954 *Subprogram O3954 call*

N3 G90 G54 G00 X.. Y.. *Normal program start*

N4 ...

...

...

<...Machining...>

...

...

N45 M30 *Main program end*

%

To visualize processing of the two programs by the CNC system, follow all operational steps in the order of program execution. During program O3903 execution, the control system will follow the following order of operations (instructions):

Set program number O3903 as the current program number

Display comment on the display screen

Set the units of measurement (inches in the example)

Branch out to the top of subprogram O3954

Execute all blocks in the subprogram O3954

When M99 is processed, the subprogram ends and returns to the main program

The main program is processed, beginning with the block N3

When M30 is processed, the main program ends and returns to the beginning

When the CYCLE START switch is activated, steps 1 to 8 are repeated

As the example shows, main program uses increments of 1, subprogram also uses increments of 1, but starting with N101 block number. There are two reasons for it. The first reason is that a properly designed subprogram will not likely be a subject to any major changes - there should be no need to add any extra blocks into the subprogram once it has been debugged. The second reason is even more important. The lack of duplicated sequence numbers will be visible on the control display screen. Display of active block numbers will quickly inform the CNC operator whether main program or subprogram is being processed. Fanuc controls are very forgiving about block numbers and allow identification of block sequences freely, within a specified range.

To illustrate the described concept, here is an example. In a simple application, where a main program calls a single subprogram, there should be no problem in block numbering. Even if the sequence numbers are duplicated in both main program and subprogram, it is not likely there will be any confusion. On the other hand, when several subprograms are called from the same main program, duplicated block numbers appear during main program processing, as well as when subprograms are processed. Such a situation may confuse the CNC operator to the extent of losing track of what is really happening at the control system at any given time.

To avoid this problem, consider assigning *unique block numbers* to each subprogram, thus preventing a duplication. One method is to identify the subprogram numbers in the high thousands series, for example O6100, O6200, O6300, etc. Then, block numbering in a subprogram can be based on the subprogram number. For example:

O6100 (SUB 1)

N6101 ...

N6102 ...

N6103 and soon

O6200 (SUB 2)

N6201 ...

N6202 ...

N6203 and soon

This method works only with the maximum of one hundred blocks, suitable for many subprograms. CNC operator finds it easy to monitor a program with several subprograms. This is not a foolproof method for all programs, but the idea will work for most jobs and may be expanded.

Protected Subprograms

Subprograms are special programs designed to be used frequently. Special subprograms may be even stored in the system memory permanently, to be called by all or many other programs. Any interference with these subprograms, accidental or intentional, can prove to be disastrous. If only a single subprogram is lost from the memory, it may halt literally hundreds of programs that depend on the use of this ill-fated subprogram.

Fanuc controls address this potential problem by allowing an assignment of certain specified series of program numbers that can be locked by a system parameter setting. As a typical example, a section of program number series 9000 (within the range of O9000-O9999), will not show on the control display, when locked by a system parameter. Also, programs in this series cannot be edited or printed out, etc. If the locking parameter is not set, programs of the 9000 series behave normally, like any other program. In order to take advantage of this feature to protect some important programs from unauthorized editing or even viewing, consult Fanuc documentation for further details.

D. SUBPROGRAM DEVELOPMENT

Before a subprogram can be developed, it must be well thought out and planned. Since the most common application for subprograms is repetitive pattern of machining, the programmer should have the ability to *recognize* a machining pattern suitable to be used as a subprogram.

Repeating Pattern Recognition

This ability to recognize a repeating pattern is a matter of experience. The first indications come when writing a conventional program block by block. Visually scan the written copy first. If there are *repeating* clusters of consecutive blocks containing the *same* data, it is a very good reason to evaluate the program more carefully and possibly develop a subprogram.

An experienced programmer will not write a program the long way first - that is a waste of time. Programming experience enhances the ability to recognize a potential for subprograms at the early stages of program planning. However, for a programmer with limited experience, there is no damage done by developing a long program first. It takes more time and lacks efficiency. However, this is how professional experience is developed and cultivated over time. With limited experience, be willing to rewrite a program from a single long format to a main program and one or more subprograms. Programmer should be able to identify those sections of long programs that can qualify as subprograms. Once such a series of repetitive data is identified in a conventional program, it is only a matter of small adjustments to separate these repetitive clusters and define them as true subprograms.

Tool Motion and Subprograms

One of the most common sub-programming applications is a tool path machined at different locations of the part. For example, a ten hole rectangular pattern needs to be programmed for blind holes - *Figure 39-4*.

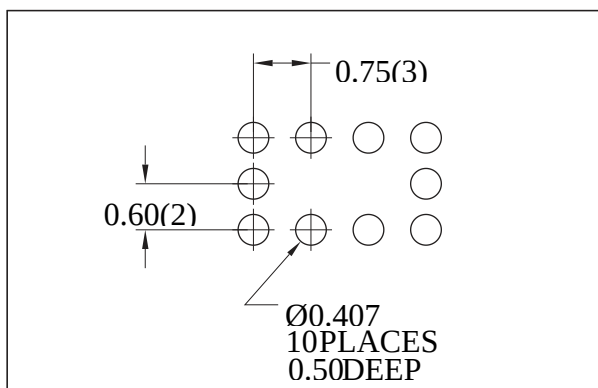


Figure 39-4

Detail of hole pattern used in program O3904

This hole pattern is repeated at four specified locations of the part, as illustrated in *Figure 39-5*.

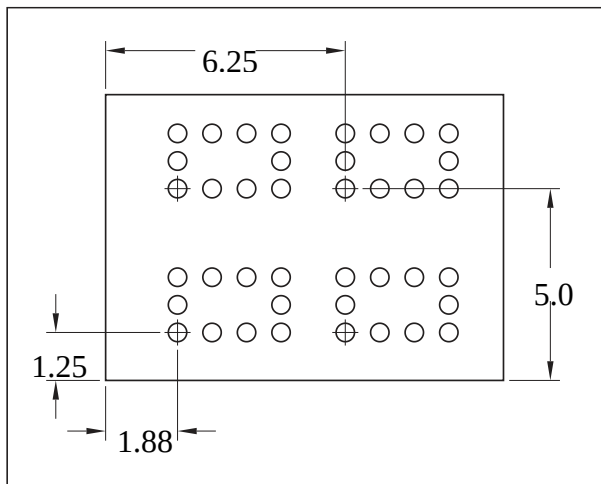


Figure 39-5

Hole pattern layout for program examples O3904 and O3905

(Both using subprogram O3955)

Subprogram O3955 contains this pattern and uses L or K address to establish the number of fixed cycle repeats. In the first main program O3904, tool motion precedes the subprogram block. To start program development, concentrate on the hole pattern. First, select G91 incremental mode for the pattern. Then program X and Y incremental motions, starting from any hole, such as the lower left hand corner, and continue in one direction - Figure 39-6.

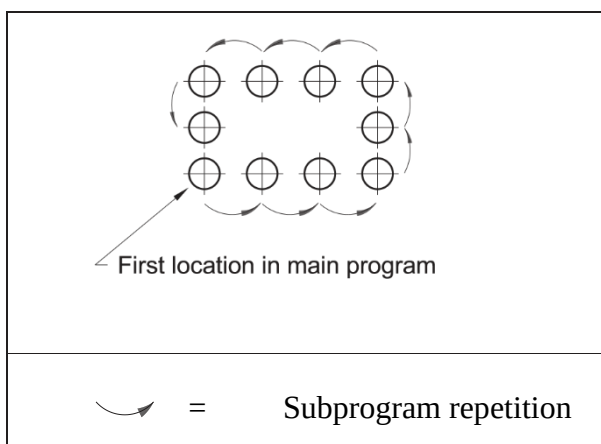


Figure 39-6

Subprogram O3955 processing flow

O3955 (SUBPROGRAM)

(FOUR-CORNER LOCATIONS)

N551 G91 X0.75 L3 (K3)

N552 Y0.6 L2 (K2)

N553 X-0.75 L3 (K3)

N554 Y-0.6

N555 M99

%

This subprogram is designed to machine *nine* holes in a rectangular pattern. The tenth hole - actually it is the first hole - is machined in a block with cycle call or rapid motion. The four pattern locations are not included in the subprogram - *they must be included in the main program*. Since

the main program is using absolute mode G90, all individual pattern locations can be established:

O3904 (MAIN PROGRAM)

(FOUR-CORNER PATTERN - DRILLING BLIND HOLES)

N1 G20

N2 G17 G40 G80

N3 G90 G00 G54 X1.88 Y1.25 N4 G43 Z1.0 S350 M03 H01

N5 G99 G81 R0.1 Z-0.6221 F3.5 (LL HOLE

**N6 M98 P3955 (LL
PATTERN)**

N7 G90 X6.25 Y1.25 (LR HOLE

**N8 M98 P3955 (LR
PATTERN)**

N9 G90 X6.25 Y5.0 (UR HOLE

**N10 M98 P3955 (UR
PATTERN)**

N11 G90 X1.88 Y5.0 (UL HOLE

**N12 M98 P3955 (UL
PATTERN)**

N13 G80 G90 G28 Z1.0 M05

N14 G91 G28 X0 Y0

...

Only one cutting tool was used for this example, other tools will follow the same programming procedure. This method of the last example is more common - in absolute mode from main program, tool is positioned at the lower left hand corner of the pattern and the first hole is drilled at that location. Then the subprogram is called, and the remaining nine holes are drilled, using incremental positioning commands and the number of cycle repetitions. Number of repetitions in the subprogram is always the number of spaces, *not* the number of holes.

A simpler way, particularly useful for a great number of pattern locations, is to *combine* the rapid motion to the pattern start point location with the subprogram call. This is acceptable for most control systems:

O3905 (MAIN PROGRAM)

(FOUR-CORNER PATTERN - DRILLING BLIND HOLES)

N1 G20

N2 G17 G40 G80

N3 G90 G00 G54 X1.88 Y1.25

N4 G43 Z1.0 S350 M03 H01

N5 G99 G81 R0.1 Z-0.6221 F3.5 M98 P3955

N6 G90 X6.25 Y1.25 M98 P3955

N7 G90 X6.25 Y5.0 M98 P3955

N8 G90 X1.88 Y5.0 M98 P3955

N9 G80 G90 G28 Z1.0 M05

N10 G91 G28 X0 Y0

...

The major advantage of O3905 is shortening the length of program O3904 - either method produces the same results, and the selection is a matter of personal preference. Note the seemingly unnecessary repetitions of modal G90 and X and Y axes. Modal values have to be followed extra carefully for subprograms.

Modal Values and Subprograms

All modal values in effect when the subprogram is called will remain in effect for that subprogram, unless changed within

In examples O3904 and O3905, note repetitions of G90, X6.25 and Y5.0. They are *very important*. Subprogram O3955 changes the control status to incremental mode G91, and the last hole of the ten hole pattern is *not* the same as the first one. The first hole of the pattern is machined when a rapid motion to that hole is completed in absolute mode. That happens in the main program, *not* within subprogram.

Here is another common problem. Finish contour subprogram uses cutter radius offset G41/G42 with D address. If the same subprogram is used for semi finishing and to leave some stock, for example, it will not work. The reason is that the D offset is fixed and is stored in the control as full cutter radius. Solution? Use *two* D offsets and remove the D address from subprogram, then call it together with M98:

M98 P.. D..

This way, the offset number D can be changed anytime the subprogram is called, without any change to the subprogram itself. This method is useful if the programmed contour requires *two* or *more different* offset settings, but it does not work on all controls. Here is the content of a simple contouring subprogram, with embedded D offset. D51 setting amount is equal to the cutter radius:

O3956 (CONTOUR SUBPROGRAM - A)

N561 G41 G01 X0 D51 F10.0 (D.. INCLUDED)

N562 Y1.75

N563 G02 X0.25 Y2.0 R0.25

N564 G01 X1.875

N565 Y0

N566 X-0.75

N567 G00 G40 Y-0.75

N568 M99

%

For contour finishing, this subprogram will be called by normal means, typically from the main program:

M98 P3956

The same subprogram can be used for finishing *as well as* for semi finishing, leaving some

material stock, but two D offsets have to be used, such as D51 and D52. In this case, offset D51 stores the amount of cutter radius and contain the stock allowance (D51 = cutter radius + stock), D52 stores the finishing radius only (D52 = cutter radius). For an end mill, the set amounts could be:

D51 = 0.250 radius + 0.007 stock = 0.257

D52 = 0.250 radius + 0.000 stock = 0.250

Next, the D.. has to be *removed* from the subprogram:

O3957 (CONTOUR SUBPROGRAM - B)

N561 G41 G01 X0 F10.0 (D.. NOT INCLUDED)

N562 Y1.75

N563 G02 X0.25 Y2.0 R0.25

N564 G01 X1.875

N565 Y0

N566 X-0.75

N567 G00 G40 Y-0.75

N568 M99

%

Control system *does* require the D offset but not necessarily in the same block as G41/G42. As long as the D address is specified *before* G41/G42, it can be passed on to a subprogram from the main program or even another subprogram, depending on operation:

M98 P3957 D51 ...for semi finishing

M98 P3957 D52 ...for finishing

This is a very powerful method of using subprograms for more than one operation, if the control system supports it.

Return from a Subprogram

All current modal values should be cleared in the main program when a subprogram is completed. Settings that may have changed in the subprogram are absolute or incremental mode, motion command, coolant and others. Subprogram is always a special *branch* of another program - it is a *continuous extension* of the program of origin, and it is its integral part. All modal values set anywhere in the program are valid until changed or canceled by a command of the same group. M99 subprogram end function will *not* cancel any modal values that are currently active.

As both O3904 and O3905 examples show, a fixed cycle is called from the main program only once. All modal cycle data are carried forward to the subprograms. Main program clearly shows current modal settings.

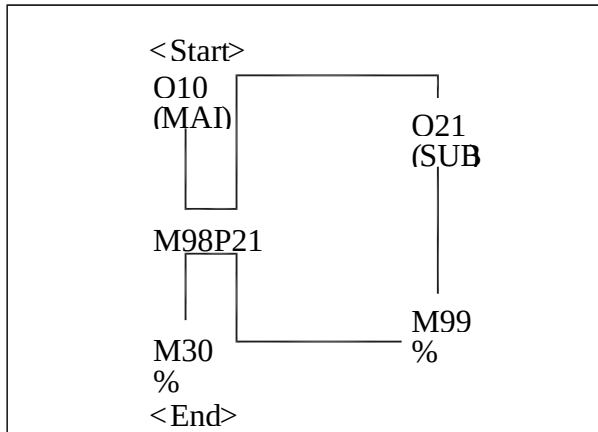
E. MULTI LEVEL NESTING

The last example has shown the main program that calls only one subprogram, and the subprogram does *not* call another subprogram. This is called *one level* nesting, or nesting at one level deep. Modern controls allow nesting up to four levels deep. That means, if the main program calls a subprogram number one, this subprogram can call a subprogram number two, that can call a subprogram number three, and that can call a subprogram number four. This is called a *four level* nesting. All four levels are rarely needed for any practical application, but these are the

programming tools available, just in case. The following examples show program processing flow of each nesting level.

One Level Nesting

One level nesting means that a main program calls only one subprogram and nothing more. Subprogram that is nested one level deep is the most common in CNC programming. Program processing starts at the top of the main



program. When a subprogram is called from the main program by M98 P.. block, the control makes a forced branch to the beginning of called subprogram and processes its contents, then it returns to the main program to process all remaining blocks of the main program - *Figure 39-7*.

Two Level Nesting

Processing of a subprogram that is nested two levels deep also starts at top of the main program. When the control encounters a subprogram call for the first level, it will branch from the main program and starts processing all blocks in the first subprogram, starting from its top. During processing of the first level subprogram, CNC system encounters a call for a second level subprogram.

At this point, processing of the first level is *temporarily* suspended and CNC system branches to the second level. Since there is no subprogram call from the second level, all subprogram blocks will be processed. Anytime the block containing M99 function is encountered, CNC system will automatically return to the program *it branched out of*. It will *resume* processing of that program, temporarily suspended before.

Return to the program of origin will normally be to the block immediately following the subprogram call block in that program. All remaining blocks in the first subprogram will be executed until another M99 function is encountered. When that happens, the control system will return to the program it branched out of (program of origin), in this case to the main program.

Since there are still some blocks left in the main program, they will be processed until the M30 function is encountered. M30 terminates the execution of the main program. *Figure 39-8* illustrates schematically the concept of a two level subprogram nesting.

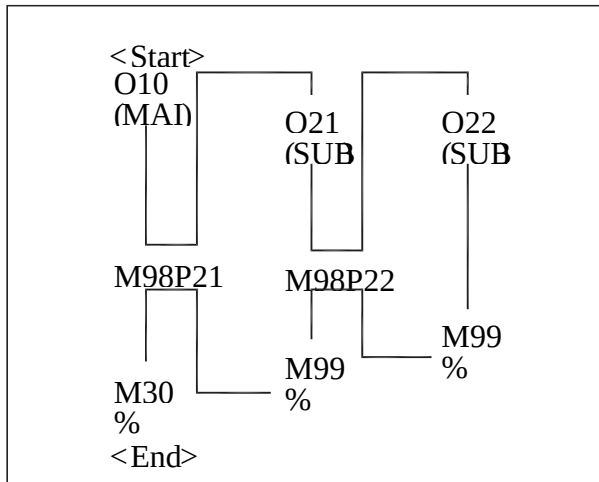


Figure 39-7

One level subprogram nesting Figure 39-8

Two level subprogram nesting

Three Level Nesting

Nesting up to three levels deep is the next logical extension of two level nesting. As before, starting at top of the main program (program O10 in the example illustrated in Figure 39-9), the first branch will be to the first level O21, another branch follows O22 and there is an additional branch to O23. Each subprogram is processed up to the next subprogram call, or the end of subprogram. Program processing will always return to the block following the subprogram call, ending in the main program.

Four level subprogram nesting

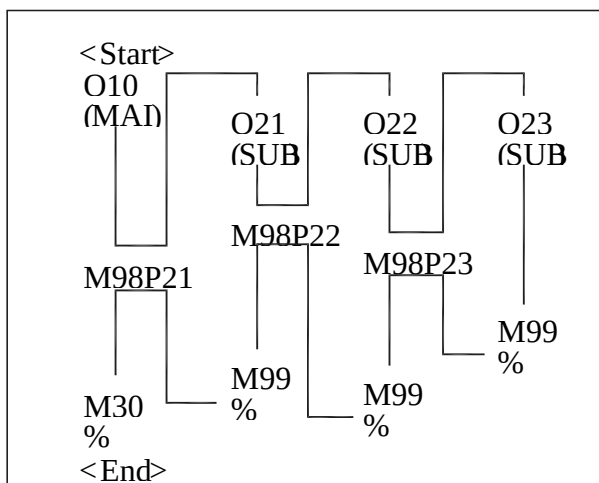


Figure 39-9 Three level subprogram nesting

Four Level Nesting

The logic of multi-level subprogram nesting should be pretty clear by now. Four level nesting is just a multiple extension of a single nesting and is logically identical to all previous examples.

Unnecessary addition of more branches for a multi depth subprogram nesting makes any programming application that much more complex and more difficult to master.

Programming subprogram nesting into the four level depth (or even the three level depth) will

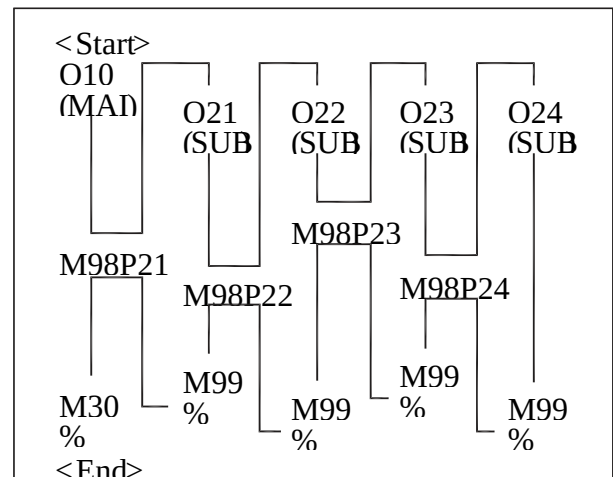


Figure 39-9

require a full understanding of program processing order - and having a suitable application for it. In a typical machine shop programming, there is seldom any need to use level three and level four nesting. If a good example of a four level nesting application is found, the typical program flow will conform to the format illustrated in *Figure 39-10*.

Nesting Applications

Considering the reality that each subprogram can be repeated up to 9999 times in any program that calls it, shows the enormous programming power available to use and explore. Always be aware of potential difficulties, even dangers, when developing subprograms with several multi nested subprograms. Such a programming approach may result in a short program, but at the cost of a long development time. Program preparation time, its development and debugging often take more time than writing conventional programs. Not only the logical development is complex and more time consuming, a significant portion of programming time must be spent on careful and thorough documentation of the process flow of all programs, setting up the initial conditions, checking validity of all data, etc.

There are many fairly experienced CNC programmers in the machining trades field, who try to use a multi-level nesting at all costs, and the more levels, the better programmers they feel they are. These programmers, more often than not, use such complex programming technique as means of expressing their so called 'professional skill', usually measured against other programmers. Often, this is nothing more than an unnecessary contest, a frustration perhaps, and definitely an expression of a little ego trip.

When a programmer becomes obsessed with making the program as short as possible, *at any and all costs*, he or she is taking the wrong track. Such programs, even if they are technically flawless and logically correct, are not always very easy to use by the CNC operator. Machine operator with limited or no programming knowledge will find these programs extremely intimidating - even skilled and experienced operators will find them hard to read, hard to interpret, and most likely, they will be unable to make any substantial changes to them, in order to modify or optimize the programs for even a better performance.

A simple general rule for multi-level nesting technique use it only in those cases, when the frequency of their future deployment justifies the extra time spent for their development. Like anything else, many nesting levels offer certain advantages and many inevitable disadvantages.

F. CONTOURING WITH A SUBPROGRAM

So far, a number of programming examples provided have been using a subprogram. They all related to machining holes and, hopefully, offered enough material to understand the concept of sub-programming (there will be one more - *a rather special one* - at the end of this chapter, so look for it). There are other examples found throughout the digital book that make generous use of subprograms.

Here is one more example relating to this chapter, this time applying a simple XY contouring work to a multiple Z depth - evaluate *Figure 39-11*.

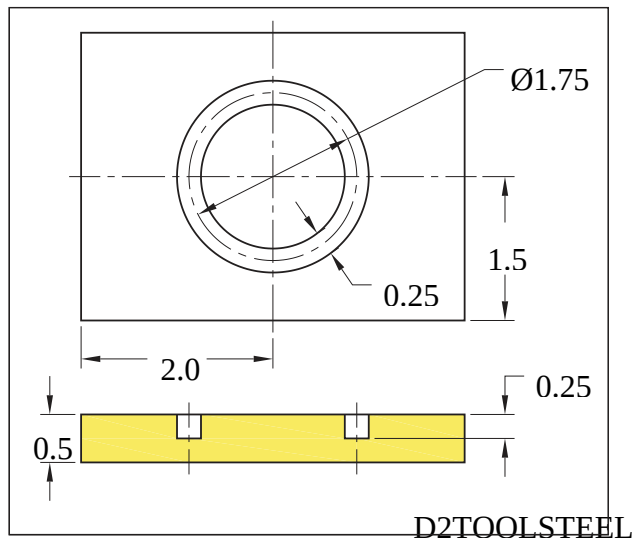


Figure 39-11

Main program O3906 using subprogram O3958

The job requires a groove with a 1.750 pitch to be machined to the depth of 0.250. It is a utility or a rough groove, so there is no need for precision tolerances, or even high quality of surface finish. All needed is a 0.250 center cutting end mill (slot drill), plunge to the depth, program a 360 circular tool path, and job is done. *Well, almost.*

Even in a material that cuts well, for example brass, splitting a single depth cut of 0.250 into two depth cuts of 0.125 may prove beneficial. Material for the example is *D2 tool steel*, rather a tough material. Tool will run at only 630 r/min and only plunge into the material 0.010 at a time, repeating

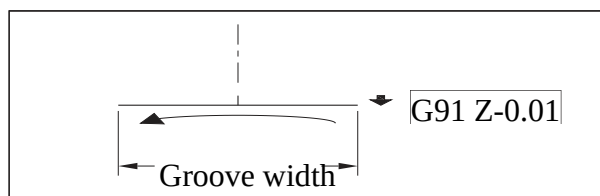


Figure 39-12

Detail of the subprogram O3958 - front view shown

the groove profile 25 times, for $25 \times 0.010 = 0.250$ total required depth. Preference for a subprogram in such a case is without a question. Symbolic detail of the depth cut for a single increment is illustrated in Figure 39-12.

Subprogram O3958 contains only *tool motions common to all groove cuts* - that means the 0.010 incremental plunge cut and the 360 circular cut. All other motions will be included in the main program O3906. Note the word *incremental* for the plunge depth. The 0.010 depth must be programmed incrementally, otherwise it will only cut at the absolute depth of Z-0.01 - *all twenty five times!* Here is the complete main program O3906, followed by a single related subprogram O3958 (T01 is assumed to be in the spindle):

O3906 (MAIN FOR SIMPLE DEEP GROOVE)

(T01 - 0.250 DIA CENTER CUTTING END MILL)

N1 G20

N2 G17 G40 G80

N3 G90 G54 G00 X2.875 Y1.5 S630 M03

```

N4 G43 Z0.1 H01 M08
N5 G01 Z0 F10.0    (START Z POSITION AT Z0!)
N6 M98 P3958 L25   (CALL SUBPROGRAM 25 TIMES)
N7 G90 G00 Z1.0 M09
N8 G28 Z1.0 M05
N9 M30 %
O3958 (SUB FOR O3906)
N581 G91 G01 Z-0.01 F0.5 (INCREMENT BY -0.01)
N582 G03 I-0.875 F2.0    (FULL CIRCLE CONTOUR)
N583 M99
%
```

Intentionally, the program as presented is simple. It does show, however, two important considerations that have to be maintained in any subprogram development. These considerations relate to maintenance of a *continuous relationship between the main program and the subprogram*. They can be described as special requirements:

... to maintain a transfer from the main program to the subprogram (before subprogram is called)

... to maintain a transfer from the subprogram, back to the main program (after a subprogram is completed)

The first requirement is met in block N5. The Z-axis position *must be at Z0*, and nowhere else! Being at Z0, it will enable the tool to increment 25 times the distance of 0.010, resulting in 0.250 groove depth. Described differently, the tool start position before a subprogram is called must be at a position that results in a correct tool path (such as depth).

The second requirement is met in block N7. It is the G90 command that makes this block special. Why? Because the subprogram uses G91 command - incremental mode. When subprogram processing returns back to the main program, it no longer benefits from incremental mode, and G90 changes the incremental mode back to absolute mode.

G. TOOL CHANGE SUBPROGRAM

Programming sequence for a typical ATC (automatic tool change) is usually short and simple. For CNC milling systems, M06 function will normally do the job and for CNC lathes, it is the T function that does the same thing. Tool change cannot be programmed without establishing certain conditions. Program functions relating to machine zero return, coolant cancellation, spindle stop and others, are all an integral part of the tool change routine. It may take three, four, five or more program blocks to establish the right conditions - *every time* the automatic tool change is programmed, which can be quite often. Even more significant is the fact that most blocks always have the same contents, regardless of the program being used.

As an example of this concept, consider the following sequence of operations - they are quite typical, required to program a tool change for several tools in a single program.

The example is based on a typical vertical CNC machining center, and uses automatic tool change function (ATC):

```

Turn off the coolant
Cancel fixed cycle mode
Cancel cutter radius offset mode
```

Turn off the spindle

Return to Z-axis machine reference position

Cancel offset values

Make the actual tool change

These seven individual operations occur in *every program* that requires this particular tool change, and they will occur for *every tool* in *each program*. That is a lot of programming for a simple tool change sequence. To make such programming easier, develop a subprogram that includes all seven operations, then call it in the main program whenever a tool change is required:

O3959 (TOOL CHANGE VERTICAL MACHINING CENTER)

N1 M09

N2 G80 G40 M05

N3 G91 G28 Z0

N4 G49 D00 H00 N5 G90 M06

N6 M99

%

This example can be easily modified for a different machine design or for a CNC horizontal machine. It may even include special requirements, such as specific manufacturer's options. Tool change may even be programmed at a certain machine table position. The only modification would be the addition of a G53X.. Y.. block before the tool change block. Another example is a special code for tool change *and* the coolant ON function. Some machine manufacturers create a special M function, combining the two standard functions, for example, M16 - which is the combination of M06+M08 standard functions.

Also note the various cancellation functions - there are quite a few of them in subprogram O3959. When designing such a subprogram, the programmer has absolutely no idea when the program is loaded, whether the coolant will be ON or OFF; no idea if a fixed cycle or the cutter radius offset is active or not. Also, the programmer has no idea as to what the current status of G90 or G91 modes is.

Their *actual* status is really not that important. These cancellations are included in the subprogram, taking advantage of the fact that a cancellation of a function that is already canceled will be ignored by the control system. As the example shows, even a 'simple' tool change sequence requires some serious thinking.

H. 100 000 000 HOLE GRID

In the last section of this chapter, perhaps a little deviation from the digital book seriousness will be tolerated. This section will look at subprograms from a different angle, but with a *real* example. The following exercise takes the sub-programming power to the very extreme. Although it is presented primarily on a light note, it does serve a very practical purpose - it shows the power of subprograms and, hopefully, makes a strong case for their use.

The example illustrates how one hundred million holes, (yes, *one hundred million* holes), can be spot drilled *and* drilled using a program of only 29 blocks for the two cutting tools. These 29 blocks even include program numbers and stop codes (% signs). *Figure 39-13* shows a simple grid pattern of 10000 rows (X) and 10000 columns (Y).

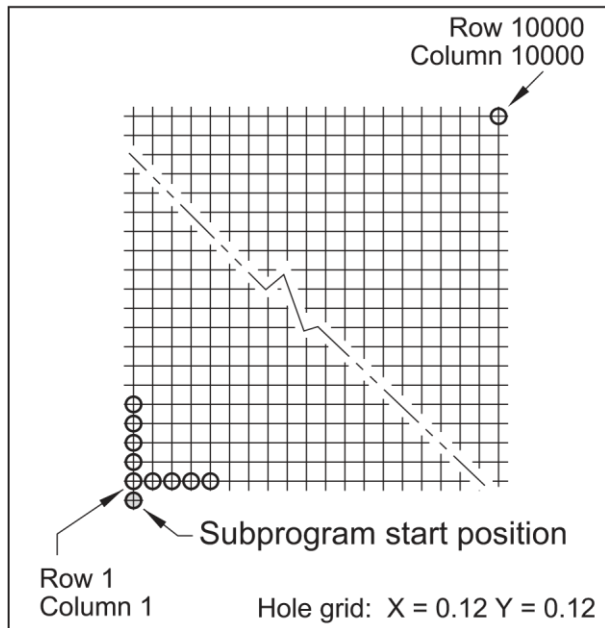


Figure 39-13

100 000 000 holes - rectangular grid pattern

To make the example reasonable, simple, and interesting at the same time, all holes are very small, only 5/64 (0.0781), with a pitch of 0.120 along each axis, resulting in a square grid pattern of holes very close to each other.

Only two tools are used, a *spot drill* with a 90° tool point angle to startup the hole for drilling and a 5/64 drill. Both cutting tools start machining from R0.06 cycle position above the plate to their respective depths: Z-0.04 for the spot drill and Z-0.215 for the drill.

From programming point of view, the program design is not difficult at all - it will use a main program and one subprogram. Programming procedure is the same for 100 000 000 holes, as if the grid were only 100 holes. The main program contains all standard settings and also calls the subprogram. This subprogram will repeat the active fixed cycle 9999 times, for *two rows*, one in each direction.

Start position for the first tool motion is at an arbitrary location X1.0Y1.0 (shifted by 0.120 along negative Y-axis). Fixed cycle drills the first hole, repeats itself 9999 times, shifts in the positive Y-axis once, drills a hole and repeats along the negative X-axis 9999 times again. This subprogram pattern repeats 5000 times - specified in the main program body:

O3960 (SUBPROGRAM)

N601 G91 Y0.12

N602 X0.12 L9999

N603 Y0.12

N604 X-0.12 L9999

N605 M99 %

O3907 (MAIN PROGRAM)

N1 G20

N2 G17 G40 G80 T01 (SPOT DRILL)

N3 M06

N4 G90 G00 G54 X1.0 Y1.0 S3000 M03 T02

N5 G43 Z1.0 H01 M08
N6 G99 G82 R0.06 Z-0.04 P30 F5.0 L0
N7 M98 P3960 L5000
N8 G90 G80 Z1.0 N9 G28 Z1.0 N10 M01
N11 T02 (5/64 DRILL)
N12 M06
N13 G90 G00 G54 X1.0 Y1.0 S3000 M03 T01
N14 G43 Z1.0 H02 M08
N15 G99 G81 R0.06 Z-0.215 F4.0 L0
N16 M98 P3960 L5000
N17 G90 G80 Z1.0
N18 G28 Z1.0
N19 G91 G28 X0 Y0
N20 M30
%

This program design takes an advantage of the subprogram nesting and the maximum number of repetitions.

What makes the program even more interesting is the estimate of *machining time*. This may go a little too far, but let's finish the fun. Before reading the whole page, make a guess - *how long will it take to machine all holes with the two tools?* Both speeds and feeds are reasonable for most materials, so are the clearances and dwell time for spot drilling. Rapid traverse of 475 in/min is assumed in all axes, not unreasonable. It is worth the few calculations? Motions between the machine zero and the first location are disregarded in both directions for convenience.

The first calculation finds the time it takes to make a rapid motion between all holes. One hundred million spaces (less one initial space), multiplied by 0.12 divided by 475 in/min is 25,263.1576 minutes. These motions will be multiplied by two, for two tools, therefore 50,526.3153 minutes.

Spot drill will move 0.060 from clearance to the top of part and another 0.040 depth of cut, for the total length of 0.100, multiplied by one hundred million holes at the rate of 5.0 in/min, therefore cutting time for spot drilling will be 2,000,000 minutes. Spot drill will rapid out of the hole one hundred million times the distance of 0.100, at the rate of 475 in/min, totaling 21,052.6316 minutes; the dwell time at each location is 0.030 seconds - translated into minutes, it will take another 50,000 minutes.

Actual drilling will take place to the depth of 0.215 from 0.060 clearance level, for the total travel of 0.275 at the rate of 4.0 in/min - which is another 6,875,000 minutes. Drill will rapid out of one hundred million times by the distance of 0.275, at the rate of 475 in/min, adding another time of 57,894.7368 minutes.

The grand total of all results is 9,054,473.6837 minutes, which is 150,907.8947 hours, which is 6,287.829 days, which is 17.2269 *years*. Believe it or not, it will take more than seventeen years of *uninterrupted* machining, to spot drill and drill one hundred million holes - and all that can be done with the main program and a subprogram totaling just over two dozen blocks of input.

Going into related details, size of the plate without margins would have to be 100 100 feet, so the actual machine travel would have to be greater than 100 feet along the X-axis *as well as* the Y-axis. Hardly any CNC machine on the market can handle this monstrous task. How would the

plate be mounted, for example? That is another question.

To make the example even more fun for one last time, consider the time spent on programming, doing it *without* a subprogram and *without* the repetition count (L/K). Assuming that each block will take 6 seconds to write, and 55 blocks will fit on a standard paper (hard copy), it would take about 19 years (yes, *nineteen years!*) just to write the program for the two tools (no interruptions, of course). As far as the paper is concerned, it would end up with 'only' 1,818,182 sheets, or a stack of approximately 705 feet (215 meters) thick. Enough of that - ***subprograms do work.***

32. DATUM SHIFT

The majority of CNC programs will be programs for a single job - a job that is relative to a specific machine available in the shop. Such a particular job will have its unique characteristics, its special requirements as well as its own toolpath. Toolpath is the most important of all features of a typical CNC program.

It is the CNC programmer's main responsibility to develop a functional toolpath for any given job, without errors and in the most efficient way. Toolpath development is very important, because it represents a machining pattern unique to the job at hand. In most programming jobs, this machining pattern is executed for the given job only and is irrelevant to any other CNC program. Often, programmers encounter opportunities, where an existing machining pattern can be used for many new jobs. This discovery will encourage development of all programs more efficiently and produce CNC programs for many additional applications and without errors.

Programming method that addresses this issue is known as *Translation of a Machining Pattern* or, more commonly, a *Datum Shift*. The most typical example of this method is a temporary change of program reference point (program zero) from the original position to a new position, so called *work shift*. Other programming methods include *Mirror Image*, described in the next chapter, *Coordinate Rotation* and *Scaling Function*, described in the chapters that follow.

This chapter describes in detail the advanced subject of *Datum Shift*, also known as *Machining Pattern Translation*. This is a basic feature of all CNC systems that can be applied in a variety of ways.

A. DATUM SHIFT WITH G92 OR G50

In essence, datum shift is a temporary or permanent relocation of the part zero (program reference point) inside of active program. When this programming method is used, it relocates an existing machining pattern (toolpath) in the program at different locations within the CNC machine work area.

In an earlier section (*see page 117*), explanation of G92 (milling) and G50 (turning) commands was covered. Review these commands now, before continuing further. In particular, recall that these commands *do not* cause any direct tool motion, but they do influence any tool motion that *follows* it. Also keep in mind that the position register command G92 and G50 registers the *absolute* coordinates of the current tool position and have no influence whatsoever on incremental dimensions, when using the G91 command for milling or U/W axes for turning. Its normal purpose is to 'tell' the control system the *current tool position*. This step is necessary at least once at the beginning of each tool to establish the relationship between the fixed program zero (part origin) and the actual position of the cutting tool. For example,

G92 X10.0 Y6.5

is 'telling' the control system that the cutting tool is set at positive 10.0 units away from program zero in the X-axis and positive 6.5 units away in the Y-axis.

What happens if a wrong position is registered? What if the preset values in G92 or G50 statement do not accurately reflect the *true*, the physical position of a cutting tool? As may be expected, the toolpath will occur at the wrong place and the result is quite likely a scrap of the machined part, tool breakage, even a damage to the machine itself. Certainly not a desirable situation.

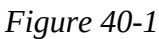
An imaginative CNC programmer always tries to find ways and special methods that take advantage of the available programming tools. G92 and G50 commands are only two of many tools that offer a tremendous power to a creative CNC programmer. Although still available, they are considered obsolete for practical purposes.

For simple jobs, there is no need for special or creative manipulations. It is not very economical

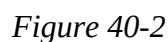
Program Zero Shift

If the G92 command is used on machining centers or the G50 command for lathes at all, rather than the more current and very efficient G54 to G59 work offsets, only *one* G92 (G50) position register command is needed *for a single tool* - assuming that work offsets are not used.

To illustrate the concept of program zero shift, a simple but relevant drawing will be used. This drawing is illustrated in *Figure 40-1*.



Based on this drawing, the four holes will be machined at *two independent* locations of the machine table setup, as illustrated in *Figure 40-2*.



G92 X(A) indicates the X distance from part zero of *Part A* to machine zero, G92 Y(A) indicates the Y distance from part zero of *Part A* to machine zero. Note that the distances are *from* program zero *to* machine zero. They could terminate anywhere else if necessary but *must* start from part zero. In order to use G92, the distances between both parts *must be known*. Simple values are used to simplify the example:

Part B: **x-11.2 y-9.7 z0** from Part A

Also note that the Z value is the *same* for both *Part A* and *Part B*, because the same tool is used for both parts. To spot drill the four holes at two locations, part program may be written this way - program O4001:

O4001

(G92 USED FOR TWO TABLE LOCATIONS)

N1 G20 G90

N2 G92 X22.7 Y19.5 Z12.5(TOOL AT MACHINE ZERO)

N3 S1200 M03

N4 M08

N5 G99 G82 X2.5 Y1.5 R0.1 Z-0.2 P200 F8.0

N6 X6.75

N7 Y5.0

N8 X2.5 (TOOL AT LAST HOLE OF PART A)

N9 ~~G80~~ Z1.0

N10 ~~G92 X-8.7 Y-4.7~~ (SET AT LAST HOLE OF A)

N11 G99 G81 X2.5 Y1.5 R0.1 Z-0.2 P200

N12 X6.75

N13 Y5.0

N14 X2.5 (TOOL AT LAST HOLE OF PART B)

N15 ~~G80~~ Z1.0

N16 ~~G92 X-9.8 Y-4.8~~ (TOOL FROM MACHINE ZERO) N17 G00 Z12.5 M09

N18 ~~X0 Y0~~ (TOOL AT MACHINE ZERO)

N19 M30

%

Several blocks require clarification, namely blocks N2, N8, N10, N14, N16 and N18. Each of them relates to the current tool position in some way. *Be very careful here.* Not understanding the principles behind G92 calculations have caused programmers many troubles in the past.

Cutting tool starts from the machine zero position for each program execution. It is also mounted in the spindle before machining. In block N2, the part zero (reference point) for *Part A* is established. Cutting tool at this point is 22.7 inches from program zero along the X-axis, and 19.5 inches along the Y-axis. The coordinate setting in block N2 reflects this fact. In blocks N7 and N8, the tool has completed the last hole of *Part A* (at X2.5Y5.0) of the *current* G92 setting.

The next critical block is N10. At this point in the program, *Part A* is completed, but *Part B* has not yet been started. Think a little now and see where exactly the tool is *after* executing block N9. It is at the position of X2.5Y5.0 of *Part A*. If the tool has to move to the first hole of *Part B*, which is also the position of X2.5Y1.5, the program has to 'tell' the control where the tool is at that exact moment - but in relation to *Part B*! That is done by a simple arithmetic calculation:

$$\text{G92 (X)} = 11.5 + 2.5 - 22.7 = -8.7$$

$$\text{G92 (Y)} = 9.8 + 5.0 - 19.5 = -4.7$$

Evaluate *Figure 40-3* to visualize the calculation. Direction of arrows in the illustration is

important for determining the axis sign in the G92 block.

Blocks N13 and N14 contain coordinates for the last tool location of *Part B*. From the illustration, it should be easy to understand meaning of the coordinate values in block N16. In order to complete the program, the cutting tool has to return to the home position (machine zero). This return

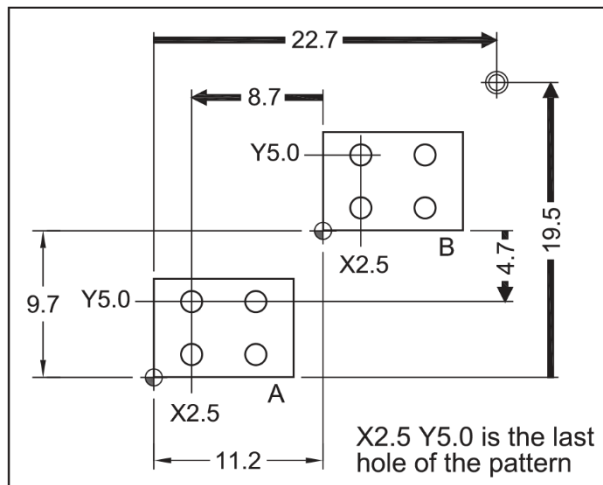


Figure 40-3

Calculations of G92 coordinates (XY) for program example O4001

will take place from X2.5Y5.0 of the *Part B*, which is 9.0 inches from machine zero along the X-axis and 4.8 inches along the Y-axis:

$$\text{G92 (X)} = 11.2 + 2.5 - 22.7 = -9.0$$

$$\text{G92 (Y)} = 9.7 + 5.0 - 19.5 = -4.8$$

Both programmed coordinates X and Y will be negative.

Once the current tool position is set at the last hole of *Part B*, return to machine zero can be made. This return is necessary, because it is the location of the *first* tool. The target position for machine zero is X0Y0 not because it is a machine zero, but because the G92 coordinates were measured from there! The actual X and Y motion to machine zero is programmed in block N18.

B. LOCAL COORDINATE SYSTEM

The G92 command for position register is as old as absolute programming itself. In time, it has been supplemented by additional commands that control the system of coordinates. Work coordinate system (G54 to G59 work offsets) has been discussed and a suggestion made that G92 should not be used when any work offset is in effect. Such a situation prevents changing the program zero on the fly, when needed only temporarily. Fortunately, there is a solution in the form of a programmable *subset* of work coordinate system (work offsets) called the *local coordinate system* or the *child coordinate system*.

There are many cases, when a drawing is dimensioned in such a way that the work offsets G54 to G59 become somewhat impractical. A good example is a bolt hole pattern. If the overall machined component is round, chances are that the program zero will be selected at the *center* of the bolt hole pattern, which offers a certain benefit in calculations.

However, if the bolt pattern is within a rectangular area, for example, part zero maybe at the *edge* corner of the part.

Normally, absolute locations of the bolt holes will have to be calculated from program zero, unless either a *shift* of the program zero is used (using G92 described earlier), or a special coordinate system is selected.

When working with work offsets, three programming methods are available to make the job a lot more convenient and perhaps even less prone to miscalculations:

Use the bolt circle center as program zero. This will be convenient for CNC programmer only, as it causes more work during setup

Use two different work offsets in the program, for example, G54 for reference to the part edge and G55 for reference to the center of the bolt circle pattern

Use local coordinate system, within the current work coordinate system (work offset) selected at the beginning of program

In all cases, one significant advantage has been gained the programmer uses calculations relating to the bolt circle center coordinates, *directly in the CNC program*, without any need of extra additions and subtractions. This method may even simplify setup on the machine. Which method is better to select and when is addressed next.

The first method, programming to the bolt circle center, is a common method and no comments are necessary.

The second method, using changes from one work offset to another, is also quite common. Its usage is not difficult. One limitation of this method is the reality that only six work offsets are available as a standard feature on typical Fanuc controls - G54 to G59. If all six offsets are needed for some work, none is left as a 'spare', to use for situations such as a bolt circle pattern. There are additional work offsets available as an optional feature of the control system.

The third method, using *local coordinate system* method, has the main advantage that it allows the use of a *dependent* - also called a *child* - coordinate system *within* the current work offset - also called the *parent* work offset. Any number of local coordinate systems can be defined within any parent work offset. Needless to say, work is always done in one coordinate system at a time. **Note** ...

Local coordinate system is not a replacement for, **but an addition to**, the work coordinate system

Local coordinate system is a supplement, or a subset, or a 'child' of the current work offset. It must be programmed only when a standard or additional work offset has been selected. There are many applications that can take advantage of this powerful control feature.

G52 Command

What exactly is the *local coordinate system*, and how does it work? Formally, it can be defined as a system of coordinates *associated* with the active work offset. It is programmed by the preparatory command G52.

G52	Local coordinate system

G52 command is always complemented by the actual known - work coordinates that set a new, that is *temporary*, program zero as illustrated in *Figure 40-4*.

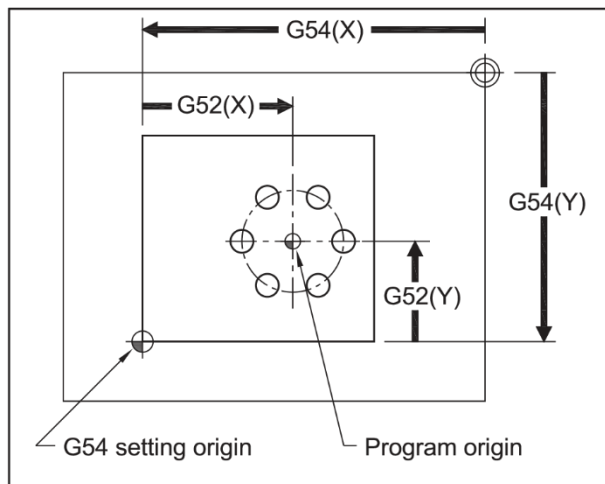


Figure 40-4

Local coordinate system definition using the G52 command

The illustration shows a bolt circle of six holes located in a rectangular plate. Typical program zero is at the lower left corner of plate and the bolt circle center is located X8.0 and Y3.0 inches from that corner, which will become the G52 shift amount. The bolt circle shown is 4.5 inches and the first hole is at 0 orientation. Subsequent holes are machined in the CCW direction as holes 2, 3, 4, 5 and 6.

What the program will do is to temporarily transfer the part zero from its current corner of plate to the bolt circle center, *in the program*. Using the illustration as a guide, follow the programming blocks, as they relate to bolt circle and in the logical order they would appear in a program:

```
G90 G54 G00 X8.0 Y3.0    (BOLT CIRCLE CENTER)
(-- WORK COORDINATE SYSTEM POSITION -----)
G52 X8.0 Y3.0
(-- NEW PROGRAM ZERO ESTABLISHED -----)
(G81) X2.25 Y0 (HOLE 1 LOCATION FROM NEW ZERO)
(-- COORDINATES FROM NEW ZERO -----)
G52 X0 Y0
(-- CANCEL LOCAL OFFSET AND RETURN TO G54 ---)
```

The modal G52 command is active until it is canceled in the program. To cancel local coordinate system and to return to the previously active work offset mode, all that has to be done is to program zero axis settings with G52:

```
G52 X0 Y0 ... last example
```

All tool motions that follow the cancellation will be relative to the original work offset, which was specified by the G54 selection earlier in the example.

Bolt circle program shown uses the techniques described. Think about the benefit of this type of programming, as opposed to letting the lower left corner be the only part zero.

First, a possible error by the CNC operator during setup has been minimized. True, the operator still has to set the G54 work offset reading at the lower left corner of the plate but does not have to do any adjustments for the bolt circle center. Programming is also easier, because the coordinate values of the bolt circle originate from the center of the bolt circle, not from the plate corner.

O4002 (G54 AND G52 EXAMPLE)

N1 G20
 N2 G17 G40 G80 T01
 N3 M06
 N4 G90 G54 G00 X8.0 Y3.0 S1200 M03 T02 (CNTR)
 N5 G43 Z1.0 H01 M08
 N6 G52 X8.0 Y3.0 (TEMP PRG ZERO AT BC CNTR)
 N7 G99 G82 R0.1 Z-0.2 P100 F10.0 L0 (NO HOLE)
 N8 X2.25 Y0 (HOLE 1)
 N9 X1.125 Y1.9486 (HOLE 2)
 N10 X-1.125 (HOLE 3)
 N11 X-2.25 Y0 (HOLE 4)
 N12 X-1.125 Y-1.9486 (HOLE 5)
 N13 X1.125 (HOLE 6)
 N14 G80 Z1.0 M09
 N15 G52 X0 Y0 (RETURN TO G54 SYSTEM)
 N16 G28 Z1.0 M05
 N17 M01
 N18 T02
 N19 M06
 N20 (...Machining continues...)

C. MACHINE COORDINATE SYSTEM

So far, the *work coordinate system*, using G54 to G59 work offsets, have been discussed, as well as the *local coordinate system* G52. They are both very powerful and extremely useful programming tools. Fanuc control system offers yet another coordinate system, not commonly used. It may be called the *third coordinate system*.

Selection of this coordinate system is exclusively with the machine coordinates and preparatory command G53.

G53	Machine coordinate system

Machine coordinate system G53 uses the coordinates measured from machine zero as an input - *always!*

At first, benefits in using this unique coordinate system may not be too apparent. Before jumping to conclusions, evaluate the rules for machine coordinates system, perhaps some applications will become clear:

Command G53 is effective only in the block where it is specified

Programmed coordinates are always relative to machine zero position

G53 is only used in the absolute mode (G90)

Current work coordinate system (work offset) is not canceled by G53 command

Cutter radius offset should always be canceled prior to G53 command

At least one possible usage emerges from these rules.

Machine coordinate system can be used to guarantee tool changes at the *same machine table location* every time automatic tool change is programmed, regardless of which work is on the table and which work offset is active. This can be applied to a single program, or as a standard for all programs for a particular machine tool. Remember, the tool change position will always be determined by the actual tool distance from the machine zero position, *not* the program zero and *not* from any other position. On many machines, or during complex setups, it is advisable to establish a fixed tool change position, regardless of the part position. A good example is machining with a rotary table, or any other permanent or semi-permanent fixture located on the machine table.

The following program illustrates the use of G53 command. It makes the tool change at a fixed position of the machine table, position that is *not* directly related to any particular program or any specific job - see *Figure 40-5*.

O4003 (G53 COMMAND USAGE - LARGE MACHINE)

N1 G20

N2 G17 G40 G80 T01

N3 G91 G28 Z0

N4 G90 G53 G00 X-170.0 Y-50.0 (TOOL CHG POS)

N5 M06 (ACTUAL TOOL CHANGE)

N6 G54 G00 X26.0 Y25.0 S1000 M03 T02

N7 G43 Z1.0 H01 M08

N8 G99 G82 R0.1 Z-0.2 P100 F8.0

N9 X53.0 Y13.0

N10 G80 G28 Z1.0 M05

N11 G53 G00 X-170.0 Y-50.0 (TOOL CHANGE POS)

N12 M01

N13 T02

N14 M06 (ACTUAL TOOL CHANGE)

M15 G90 G54 G00 X53.0 Y13.0 S780 M03 T03

N16 G43 Z1.0 H02 M08

N17 G99 G81 R0.1 Z-0.836 F12.0

N18 X26.0 Y25.0

N19 G80 G28 Z1.0 M05

N20 G53 G00 X-170.0 Y-50.0 (TOOL CHANGE POS)

N21 M01

N22 T03

N23 M06 ... (ACTUAL TOOL CHANGE)

%

The fourth item of the rules mentioned earlier states that the current work offset is *not* canceled by any active machine coordinate system command (G54-G59 work offset). Since the programming example O4003 does not illustrate this situation, the following sequence of tool motions (*not related to program O4003*) shows that G53 command is independent from G54 command:

N1 G21 (METRIC)

...

N250 G90 G54 G00 X17.7 Y35.3

N251 G01 Z-5.0 F200.0 N252 G00 Z500.0

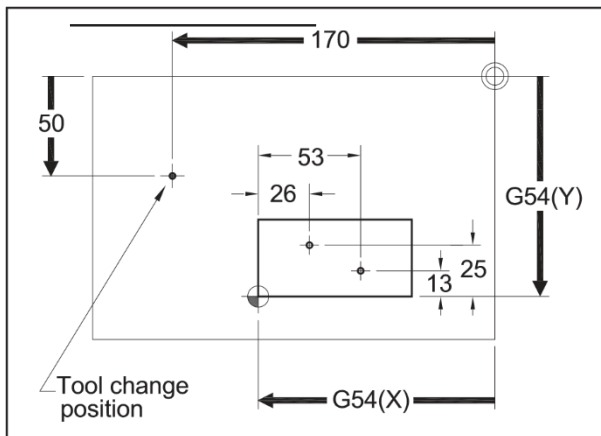


Figure 40-5

Machine coordinate system G53 - program example O4003

N253 G53 X-400.0 Y-100.0 (FIXED POSITION)

N254 M00 (MANUAL TOOL CHANGE)

N255 S1200 M03

N256 X50.0 Y35.0 (IN ORIGINAL WORK OFFSET)

N257 (...Machining continues...)

Programmed machining sequence is quite simple. Cutting tool moves to the XY part position in block N250, performs the required machining operation, such as drilling to depth in N251, rapids to a clear Z-position in N252, *then* moves to the fixed tool change position in block N253. In program stopped mode, CNC operator changes the tool manually in block N254, then the spindle speed and rotation are re-established in N255. In block N256 only the X and Y coordinate positions are specified. All other values are default values, including G54 work offset command. The previous block N256 has the same meaning as:

N256 G90 G54 G00 X50.0 Y35.0

Adhere to a good programming practice and always program a complete block that contains all setting information and do it for each new tool called. There are other practical uses for machine coordinate system, waiting to be discovered.

D. DATA SETTING

In a small or medium machine shop, job shop, or any other environment where standalone CNC machines are used, machine operator typically stores all offset settings that have to be input into the CNC system during job setup. This common method is very useful when CNC programmer

does not know actual setting values of various offsets at the time of program development, which is normal.

In a controlled manufacturing environment, for example agile manufacturing or very large volume production, this method is very costly and inefficient. An agile or large volume production uses modern advanced technology, such as CAD/CAM systems for design and toolpath development, concept of cells, robots, preset tools, automatic tool changing and tool life management, pallets, programmable auxiliary equipment, machine automation, and so on. In such an environment, there cannot be any unknown elements - relationships of all reference positions are always known and the need for offsets to be found and set at each individual machine is eliminated. All offset amounts to be set must be always known to the programmer, *before* the actual machine and tool setup take place.

There is an advantage in such information being known the offset data can be included in the program and be channeled into appropriate registers through regular program flow. There is no operator's interference and machining is fully automated, including the maintenance of tools and related offsets. All offsets are under constant program control, *including their updates* required for position changes and changes in tool length or radius.

All this high tech automation is possible with an optional control feature called *Data Setting*. Many controls have this special feature available, a feature that should never be underestimated. Even a small shop with only one standalone CNC machine can benefit from Data Setting feature, provided it is supported by the control system.

Data Setting Command

To select data setting option and to set offset data through the program, Fanuc offers a special G-command:

G10	Data setting

In its basic form, the preparatory command G10 is a *non-modal* command, valid only for the block in which it is programmed. If it is needed in any subsequent blocks, it has to be repeated in that block.

G10 command has a simple format that is different for machining centers and lathes. Be prepared to encounter minor differences in format for various Fanuc controls, although the programming methods are logically the same. Formats also vary for different types of offsets, for example, work offsets as opposed to tool length or radius offsets.

Examples in this section are for typical Fanuc controls and have been tested on a common milling and turning control.

Coordinate Mode

Selection of the absolute or incremental programming mode has a great impact on the offset values input throughout the CNC program. Regardless of which type of offset is entered along with G10 command, the programmed offset amount will *replace* the current offset amount stored in the control, if the program is in absolute mode - G90 for milling controls and XZ for turning controls.

In G91 incremental mode for milling controls and UW axes for turning, the programmed offset amount does *not replace* but *update* the offset amount stored in the control:

G10 in G90 mode = offset amount will be REPLACED

G10 in G91 mode = offset amount will be UPDATED

G90 or G91 can be set anywhere in the program, as long as the block containing the selected command is assigned *before* the G10 data setting command is called.

All three types of available offsets can be set through the program, using G10 command:

Work offsets G54 to G59 and G54.1 P..

Tool length offsets G43 and G44

Cutter radius offsets G41 and G42

This group includes all associated offsets, if available.

E. WORK OFFSETS

Before studying this section, review the concepts of work offsets in detail - *see page 127*.

Standard Work Offset Input

The standard six work offsets G54 to G59 are available for both milling and turning controls. Due to machining requirements, they are typically associated with milling controls. Programming format is the same (based on axes):

G10 L2 P.. X.. Y.. Z.. *Machining centers =mills*

G10 L2 P.. X.. Z.. *Turning centers =lathes*

Group L2 is a *fixed offset group number* that identifies the input as work offset setting. The P address in this case can have a value from 1 to 6, assigned to the G54 to G59 selection respectively:

	P2	=	
P1 = G54	5		P3 = G56
P4 = G57	P5	=	P6 = G59
	8		

for example,

G90 G10 L2 P1 X-450.0 Y-375.0 Z0

inputs X-450.0Y-375.0Z0 coordinates into the G54 work offset register (all examples for this section are metric).

G90 G10 L2 P3 X-630.0 Y-408.0

inputs X-630.0Y-408.0 coordinates into the G56 work coordinate offset register. Since the Z-amount has not been programmed, the current amount of the Z-offset is retained.

L2 = Standard work offsets

Additional Work Offset Input

In addition to the standard six work offsets for milling controls, Fanuc offers an optional set of additional offsets, G54.1P1 to G54.1P48. G10 command can also be used to input offset amounts to any one of the 48 additional work offsets while the command itself is very similar to the previous one:

G10 L20 P.. X.. Y.. Z..

Only the fixed offset group number has changed to from L2 to L20, which only selects the additional work offsets.

L20 = Additional work offsets

External Work Offset Input

Another offset that belongs to the work coordinate system group is called *External* or *Common*. This special offset is not programmable - it cannot be programmed with any standard G-code and is only used to update all work offsets globally, affecting all work offsets in a program.

To input offset settings into the external offset, G10 uses the L2 offset group and P0 as the offset selection:

G90 G10 L2 P0 X-10.0

will place X-10.0 into the external work offset, while retaining all other settings (the Y-axis, the Z-axis and any additional axis as well). In practice, when using the shown setting, *each* work offset used in a particular program will be shifted by 10 mm into the X-negative direction.

F. TOOL LENGTH OFFSETS

Tool length offset amount for milling controls can be programmed with the G10 command combined with the *Lxx* offset group. Depending on the *type* of control memory (*see page 261*), the L group will have different meanings.

There are three types of memory on Fanuc controls for the tool length and tool radius offsets:

Memory A - only one column for tool length offset

=> Input: Combined Geometry + Wear offset

Amounts: Amounts set by G10 L11 P.. R.. block

Memory B - two columns for tool offset

=> Input 1: Separate Geometry offset value

Amounts 1: Amounts set by G10 L10 P.. R.. block

=> Input 2: Separate Wear offset value

Amounts 2: Amounts set by G10 L11 P.. R.. block

Memory C - two columns for tool offset and two columns for radius offset

=> Input 1: Separate Geometry offset value

Used for: H-offset code

Amounts 1: Amounts set by G10 L10 P.. R.. block

=> Input 2: Separate Geometry offset value

Used for: D-offset code

Amounts 2: Amounts set by G10 L12 P.. R.. block

=> Input 3: Separate Wear offset value

Used for: H-offset code

Amounts 3: Amounts set by G10 L11 P.. R.. block

=> Input 4: Separate Wear offset value

Used for: D-offset code

Amounts 4: Amounts set by G10 L13 P.. R.. block

In all cases, the L number is an arbitrarily assigned offset group number by Fanuc and the P address is the offset register number in the CNC system. R setting is the amount of the offset to be set into the selected offset number. Absolute and incremental modes have the same effect on tool length programmed input as for work offsets (replacement in G90 mode, update in G91 mode).

As an example, for a CNC machining center, the following block will input the amount of negative 468 mm into the tool length offset register number 5 (five):

G90 G10 L10 P5 R-468.0

If the offset has to be adjusted in order to make the cut 0.5 mm less deep for the tool length offset 5, change to the incremental mode G91 and program:

G91 G10 L10 P5 R0.5

Note the G91 incremental mode. If the last two examples are used in listed order, the final offset number 5 amount will be -467.5 mm.

Older Fanuc controls were using the address L1 instead of the newer L11. These controls did not have a wear offset as a separate entry. For a compatibility with the older

controls, L1 is accepted on all modern controls in lieu of L11

Valid Input Range

On most CNC machining centers, the range of tool length offset values is limited:

± 999.999 mm	Metric Geometry offset input
± 99.9999 inches	Imperial Geometry offset input
± 99.999 mm	Metric Wear offset input
± 9.9999 inches	Imperial Wear offset input

Physical number of available offsets is also limited, depending on the control model. There is always a minimum of 32 offset numbers available. Optionally, the CNC system can have up to 999 offsets available (and even more), most of them as a special option.

G. CUTTER RADIUS OFFSETS

For the highest offset memory type C, the amount of cutter radius offset (D) may be input through the program, using G10 command with L12 and L13 offset groups:

G90 G10 L12 P7 R5.0

will input 5.000 radius value into the cutter radius geometry offset register number 7.

G90 G10 L13 P7 R-0.03

will input -0.030 radius amount into the cutter radius wear offset register number 7.

If the existing offset amount needs to be only *adjusted* or *updated*, use incremental programming mode. The last example of a wear offset will be updated by adding 0.01 mm:

G91 G10 L13 P7 R0.01 (NEW SETTING IS 0.02 MM)

Be careful with the G90 and G91 mode - remember to restore correct mode for subsequent sections of the program.

H. LATHE OFFSETS

Tool length offset does not apply to lathe controls, because of a different offset structure. G10 command can be used to set offset data for a lathe control, using this format:

G10 P.. X (U).. Z (W).. R©).. Q..

The P address is either the *geometry offset* number or the *wear offset* number to be set. Addresses X, Z and R are absolute values, addresses U, W and C are their respective incremental equivalents. No G90 or G91 mode is available, when using the standard G-codes of the *A Group*.

To tell apart geometry and wear offsets, the geometry offset number must be increased by arbitrary 10000:

P10001 will be geometry offset number 1

P10012 will be geometry offset 12 ... *and so on*

If the setting of 10000 is not added, the P-number will then become the wear offset number.

Here are some typical examples of offset data setting for a CNC lathe, along with expected results. All examples are *consecutive*, based on the order of input listed:

G10 P10001 X0 Z0 R0 Q0

... clears all geometry offset for *G 01* settings

(Geometry offset register 1)

G10 P1 X0 Z0 R0 Q0

... clears all wear offset for *W 01* settings

(Wear offset register 1)

Note - *Q0* also cancels setting of tool tip number in *G 01*

G10 P10001 X-200.0 Z-150.0 R0.8 Q3

... sets the contents of *G 01* geometry offset to:

X-200.0 Z-150.0 R0.8 T3

... also sets T3 in the wear offset - *automatically!!!*

G10 P1 R0.8 *Current T setting assumed*

... sets R0.8 amount in *W 01* wear offset

Note, that it may be much safer to program:

G10 P1 R0.8 Q3 *Current setting not assumed*

G10 P1 X-0.12

... wear offset *W 01* is set to X-0.12, *regardless* of its previous setting

G10 P1 U0.05

... updates X-0.12 by +0.05, to the new value of X-0.07

Note that the tool tip number (programmed in G10 application as the Q entry) will always change both geometry and wear offset simultaneously, whatever the amount or the offset type is. The reason is a control built-in safety that attempts to eliminate data entry error.

I. MDI DATA SETTING

Programming various offset settings through part program data requires full understanding of the input format for any particular control system. It is too late when an incorrect setting causes a damage to the machine or part.

One method that can be used to make sure the offset data setting is correct, is a simple test. Test all G10 entries in MDI mode on the CNC unit first, and check the results:

Select Program mode

Select MDI mode

Insert test data

For example, enter:

G90 G10 L10 P12 R-106.475

Press INSERT

Press CYCLE START

To verify input accuracy, check the tool length offset H12 - it should have the stored amount of -106.475.

While still in the MDI mode, insert another test data, for example:

G91 G10 L10 P12 R-1.0

Press INSERT

Press CYCLE START

Again, to verify, check the setting of tool length offset H12 - now, it should have the new amount of -107.475.

Develop other similar tests to follow the same routine. It is always better to start a program with confidence.

J. PROGRAMMABLE PARAMETER ENTRY

This section covers yet another aspect of programming the G10 command - this time as a *modal* command. It is used to change a system parameter, through the program. This command is sometimes called the '*Write to parameter function*' and is definitely not very common in daily programming. Inexperienced programmers should skip this section altogether. It is very important to understand the concept of control system parameters, otherwise this section will not help much. Authorization to change parameters for the machine tool, regardless of other professional qualifications, is equally important to apply this section.

W A R N I N G!

Incorrect setting of CNC system parameters may cause irreparable damage to the CNC machine!

Typical uses of this command are common to changes of machining conditions, such as spindle and feed rate time constants, pitch error compensation data, etc. This command usually appears in optional *Custom Macros* (applied by the G65 command) and its purpose is to control certain machine operations. Concept and explanation of *Custom Macros* is not covered by this digital book. A separate publication by the author is available - *see introduction*.

Modal G10 Command

When the G10 command was used for offset data setting earlier, it had to be repeated in each block. G10 for the offset entry can only be used as a *non-modal* command. Modern Fanuc controls also allow to do another type of change through the program - the change of *CNC system parameters* through a *modal* G10 command.

Many entries used in programs are automatically converted to a system parameter by the control system. For example, when programming G54, the set value is seen on the work offset screen. Yet, the actual storage of G54 setting takes place in a system parameter, identified by a certain parameter number. G54 work offset setting can be changed either through the offset data or through parameter change, when the parameter number must be known. Some system parameters cannot be changed as easily (and some cannot be changed at all), so the modal G10 command can be very useful for multiple changes. In fact, two related commands are required - G10 to start and G11 to cancel the setting:

G10 L50

(...data setting...)

G11

The data setting block has three entries:

G10 L50 .. P.. R.. G11

In case of a modal G10 and G11 combination, the commands have this meaning:

G10 Data setting mode

L50 Programmable parameter entry mode fixed

.. P.. R.. Data entry specification

G11 Data setting mode cancel

Between the block G10L50 and the block G11 is the list of system parameters that are to be set, one parameter per block. Parameter number uses the N-address and all data use P and R addresses. There are several types of parameter input:

Type of parameter input	Allowed input range
Bit type	0 or 1
Bit axis type	0 or 1
Byte type	0 to ± 127
Byte axis type	0 to 255
Word type	0 to ± 32767
Word axis type	0 to ± 32767
Two word type	0 to ± 99999999
Two word axis type	0 to ± 99999999

Watch the bit types of parameters - a single data number is always assigned 8 bits. Each bit has a different meaning, so exercise care when changing one bit but not another.

Word type is also called an *integer type* and the two-word type is also called a *long integer type*.

Parameters Notation

Numbering of bit type and bit axis type parameters is standard from 0 to 7 (*computers start counting from zero not from one*), from *right to left*:

Number #7 #6 #5 #4 #3 #2 #1 #0

where *Number* is the four digit parameter number and the #0 to #7 are individual bit positions - *note the order of numbering and the counting method*. Other, non-bit type parameters are input as a byte, word, or two word entries (there are axis and non-axis versions).

P Address

The P address is used only for parameters relating to axes (bit axis, byte axis, word axis and two word axis). If the parameter does not relate to an axis, the P address is redundant and does not have to be programmed.

If more than one axis is required to be set at the same time, use multiple .. P.. R.. entries between G10 and G11 see examples further in this section.

R Address

The address R is the new value to be registered into the select parameter number and must

always be entered. The valid range listed above must be observed. Note the lack of decimal points in the entries.

Program Portability

Programs containing even a single programmable parameter entry should be used only with the machine and control for which they were designed.

Use extreme care when a program that modifies system parameters is used on several machines

Parameter numbers and their meaning on different control models are not necessarily the same. The exact control model and its parameter numbers must be known during programming. For example, on Fanuc control Model 15, the parameter that controls the meaning of an address without a decimal point is number 2400 (Bit #0). The parameter that controls the same setting on Fanuc control Model 16 will use number 3401 (Bit #0).

The following examples illustrate various programmable parameter entries and have been tested on a Fanuc 16 Model B CNC control - lathe and mill version. Selected parameters are used for illustration only, not necessarily as typical applications. *Testing these parameters on the machine is not recommended!*

The first example changes the baud rate setting of an Input/Output device with RS-232 interface, if the *I/O Channel* is set to 0):

G10 L50

N0103 R10 G11

Parameter that controls the baud rate setting for a selected device has a number #103. From a table supplied by Fanuc, the R-value can be input:

Setting R-value	Description
1	50 baud
2	100 baud
3	110 baud
4	150 baud
5	200 baud
Setting R-value	Description
6	300 baud
7	600 baud
8	1200 baud
9	2400 baud
10	4800 baud
11	9600 baud
12	19200 baud

In the previous example,

G10 L50

N0103 R10 G11

4800 characters per second baud rate has been selected.

In another example, system parameter #5130 controls the chamfering distance for thread cutting cycles G92 and G76 (gradual pullout distance applicable to lathe controls only). The data type is a non-axis byte, unit of the data is 0.1 of a pitch and the range is from 0 to 127:

G10 L50

N5130 R1 G11

This program segment will change parameter #5130 to the value of 1. The chamfering amount will be equivalent to one pitch of the thread. Do not confuse byte with a bit byte is a value 0 to 127 or 0 to 255 for the byte axis type, bit is a single state only (0 or 1, OFF or ON, DISABLED or ENABLED), offering selection of only one of two options available. The word BIT is actually an abbreviation of two words:

Bit = **B**inary digit ('binary' means *based on two*)

Another example is for the entry of a two word parameter type. It will change the work offset G54 to X-250.000:

G90

G10 L50

N1221 P1 R-250000 G11

Parameter #1221 controls G54, #1222 controls G55, and so on. P1 refers to the X-axis, P2 refers to the Y-axis, and so on, up to 8 axes on model 16. Because the valid range of a long integer (two word type) is required, a decimal point cannot be used. Since the setting is in metric system and one micron (0.001 mm) is the least increment, the amount of -250.000 will be entered as -250000. The following example is *NOT* correct and will result in an error:

G90

G10 L50

N1221 P1 R-250.0 (DECIMAL POINT NOT ALLOWED) G11

Proper input is *without* the decimal point. An error condition (alarm or fault) will also be generated if the P address is not specified at all. For example,

G90

G10 L50

N1221 R-250000

G11

will generate an error condition. The next example is changed for a two axes input:

G90

G10 L50

N1221 P1 R-250000

N1221 P2 R-175000 G11

If this example is used on a lathe control, P1 is the X-axis, P2 is the Z-axis. On machining

center, the P1 is the X-axis, P2 is the Y-axis and P3 will be the Z-axis, if required. In either case, the first two axes of G54 work offset setting will be -250.000 and -175.000 respectively.

Sometimes it is necessary to set all axes to zero. This may be done with a standard offset setting:

G90 G10 L2 P1 X0 Y0 Z0 (MILLING CONTROL)

or write to a parameter, also for a milling control:

G90

G10 L50

**N1221 P1 R0 (SET G54 X
ORDINATE TO 0)**

**N1221 P2 R0 (SET G54 Y
ORDINATE TO 0)**

**N1221 P3 R0 (SET G54 Z
ORDINATE TO 0)**

G11

Bit Type Parameter

The next example is quite harmless and may be used as a test but be careful with any other parameters. Its only purpose is to set automatic block sequencing ON while entering a CNC program at the control. It also serves as an illustration of a bit type parameter and is a good example of some general thoughts and considerations that go into program preparation using programmable parameter mode.

On Fanuc 16 Model B (and most of the other models as well) is a feature that allows automatic entry of sequence numbers, if the program is entered from the keyboard. This feature is intended as a time saving device for manual entry of program data. In order to enable this feature, select the parameter that controls the ON and OFF status of this feature. On Fanuc 16 it is a parameter number 0000 (same as 0). This is a bit-type parameter, which means it contains eight bits. Each bit has its own meaning. Bit #5 (SEQ) controls state of the automatic sequence numbering (ON or OFF is the same as 1 or 0, but only a number can be input). An individual bit cannot be programmed, only the single data number of all eight bits. That means all other bits must be known in order to change one. In this example, the current setting of parameter 0 is as follows:

			SEQ			INI	ISO	TVC
0000	#7	#6	#5	#4	#3	#2	#1	#0
	0	0	0	0	1	0	1	0

Specific meaning of the other parameters is irrelevant for this example and should be maintained. Bit #5 is set to 0, which means the automatic block numbering is disabled.

The following program segment will turn on the bit #5, without changing any other bits:

G10 L50

N0 R00101010 G11

The resulting entry in the parameter screen will reflect that change:

			SEQ			INI	ISO	TVC
0000	#7	#6	#5	#4	#3	#2	#1	#0
	0	0	1	0	1	0	1	0

Note that all bits had to be written. The job is not done yet, however. Fanuc offers an additional feature - the *increment* for the numbering can be selected as well, for example, selection of 10 will use N10, N20, N30 entries, selection of 1 will use N1, N2, N3, and so on. The example will select increments of five, for N5, N10, N15, etc. The increment has to be set - yes - by another parameter number. On Fanuc 16, the parameter number that contains the automatic numbering value is #3216. This is a word type parameter, and the valid range is 0 to 9999. This parameter can only be activated by setting the bit #5 in parameter 0000 to 1. Program segment will look like this:

G10 L50

N3216 R5 G11

Once these settings are completed, there will be no need to enter block numbers in any program entered via the control panel keyboard.

Anytime the End-Of-Block key EOB is pressed, the N number will appear automatically on the screen, in the increments of five, saving keyboarding time during manual program input.

The idea behind G10 being modal in the programmable parameter entry mode is that more than one parameter can be set as a group. Since the two parameters are logically connected, a single program segment can be created, with the same final results as the two smaller segments earlier. The modal G10 command comes handy here:

G10 L50

N0000 R00101010

N3216 R5 G11

As neither parameter is the axis type, the address P was omitted. N0000 is the same as N0 and was used only for better legibility.

Note to Fanuc 15 users (Fanuc 15 system is *higher* than Fanuc 16) - the parameter number that selects whether the automatic sequencing will be enabled is 0010, bit #1 (SQN). There is more flexibility on Fanuc 15 - the starting sequence number can be controlled with parameter #0031, and the parameter number that stores the increment amount is #0032, with the same program entry styles as shown. Also on Fanuc 15, the allowable range of sequence numbers is up to 99999. This is a typical example of a difference between two control models, even when they were produced by the same manufacturer.

Effect of Block Numbers

Many programs include block numbers. It would be perfectly natural to assign block numbers to the last example:

...

N121 G10 L50

N122 N0000 R00101010

N123 N3216 R5

N124 G11

...

Will the program work? There are now *two* different N addresses in blocks N122 and N123. How does the control handle this situation? Rest easy - there will be no conflict. In case of two N addresses in a single block *within the G10 to G11 segment*, the first N address is the block number, the second one in the same block will be interpreted as the parameter number.

Always use extreme caution when working with system parameters - previous backup is strongly recommended

33. CNC LATHE ACCESSORIES

Any CNC machine can be equipped with additional accessories, to make it more functional or functional in a particular way. In fact, most CNC machines have at least some additional accessories, either as a standard equipment or as an option. Machining centers have indexing and rotary tables, pallets, right angle heads, etc. All these are complex accessories and require a certain amount of time to understand them well. Apart from live tooling capabilities (described in a separate chapter), many CNC lathes are equipped with accessories that are usually quite simple to program. Some of the most noteworthy and typical programmable additions (or features) of this kind are:

Chuck control

Tailstock quill

Bi-directional turret indexing

Bar feeder

Other features may also be programmable as options:

Parts catcher (unloader)

Pull-out finger

Tailstock body and quill

Steady rest / follower rest

Part stopper

... others as per machine design

Some of these accessories are fairly common, so it is worth looking at them in some detail and with a few examples of their programming applications.

A. CHUCK CONTROL

In manual operations, a chuck, a collet or a special fixture mounted on the headstock of a lathe normally opens and closes when CNC operator presses a foot pedal. For safety reasons, a chuck that is rotating cannot be opened, because it is protected by a special safety interlock. Another important feature of chucks is that the terms *open* and *close* depend on the method of chucking - *external* or *internal*. A key switch is generally available to select one type of chucking or the other. Figure 44-1 shows the difference.

Note that both terms *opened* and *closed* are relative to the setting of a toggle switch or a key switch, found on the machine itself, usually marked CHUCK CLOSED - that has two settings - IN and OUT.

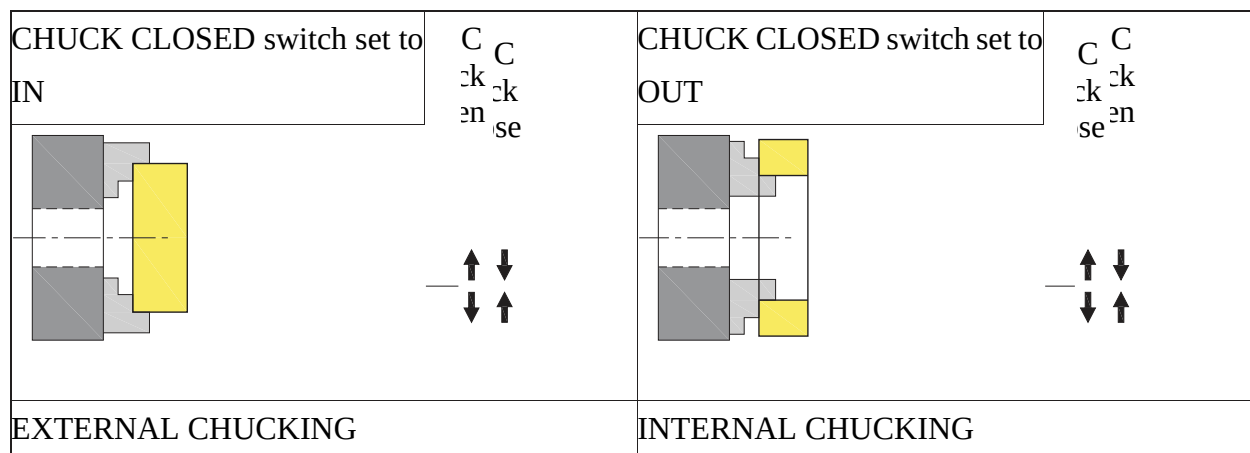


Figure 44-1

Part chucking - external and internal applications.

Note the setting of the CHUCK CLOSED switch

In some applications, such as bar feeding, it is necessary to open and close the chuck under program control. Two M functions that control the chuck or collet opening and closing are normally available.

Chuck Functions

Although the assigned numbers (normally miscellaneous functions) may vary for different machines, programming application is exactly the same. One function cancels the other. Typical M functions related to chuck control are:

M10	Open chuck
M11	Close chuck

Example:

Typical programming procedure would include spindle stop and dwell:

```
M05      (STOP  
          INDLE)  
  
M10      (OPEN  
          IUCK)  
  
G04 U0.1 (DWELL      1  
          COND)  
  
M11      (CLOSE  
          IUCK)  
  
M03      (RESTART  
          INDLE)
```

This is a very simplified sequence, in which dwell is the time required for the bar (for example) to go through to the stop position. Some bar feeders do not require spindle to be stopped to feed the bar through and others have a special programming routine of their own.

M10 and M11 functions can also be used on the machine, during setup, using MDI setting mode in manual mode. Later in this chapter, M10 and M11 functions will be used for applications associated with bar feeding.

Chucking Pressure

Amount of force required to clamp a part in the chuck is called the *chucking pressure*. On most CNC lathes the pressure is controlled by an adjustable valve, usually located in the tailstock area. Once the chuck pressure has been set, it is not changed very often. However, there are jobs that require the chucking pressure to be *increased* (tighter grip) or *decreased* (looser grip) frequently, usually within the same operation. Such special jobs will benefit from a programmable chuck pressure control.

Only a few CNC lathe manufacturers offer a programmable chucking pressure. If they do, it is in the form of two *non-standard* miscellaneous function, for example:

M15	Low chucking pressure
M16	High chucking pressure

Typically, part has to be re-clamped in the chuck before either function can replace the other,

which may disturb its position in the holding device. If chucking pressure feature is present on the lathe, read any related documentation supplied by the lathe manufacturer.

When changing chucking pressure

- manually or through the program always make sure the part is safely clamped.

Chuck Jaws

This section is not directly related to programming but covers a few tips useful to the programmer. Most chucks have three jaws, spaced 120° apart - see *Figure 44-2*.

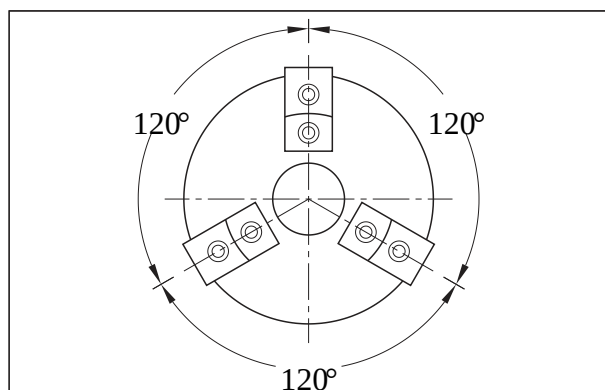


Figure 44-2

Typical three-jaw chuck for a CNC lathe

Chuck jaws may be *hard* (usually serrated for better grip) or *soft*, normally bored by CNC operator to suit the work diameter. Only soft jaws can be modified.

There is not much that can be done with hard jaws, except purchasing a suitable type for external or internal grip. Soft jaws are designed to be *bored* and the ability to do that is one of the basic skills a CNC operator must have. There are various techniques to bore soft jaws, all beyond the scope of this digital book. What is important is the understanding of what happens if jaws are not bored correctly.

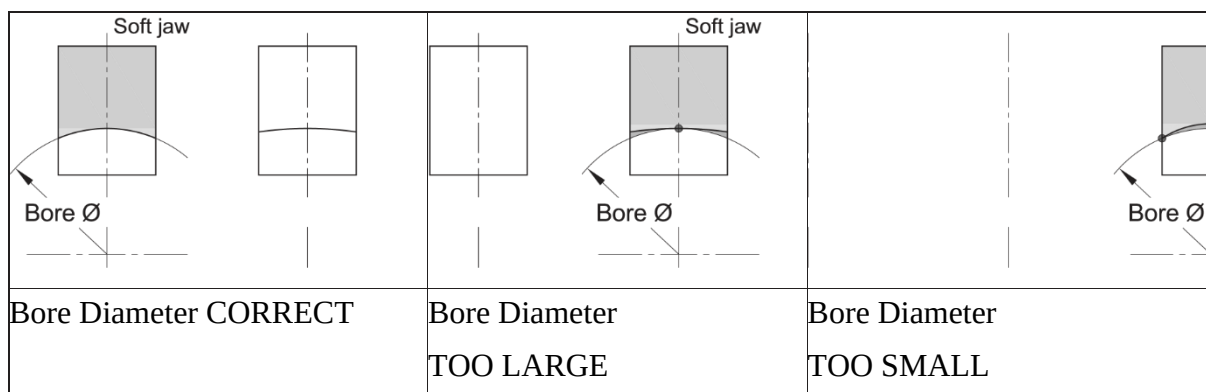


Figure 44-3

Soft jaws diameter bored correctly (left) and incorrectly

Figure 44-3 shows three versions - one correctly bored jaws and two incorrectly bored jaws. In both incorrect versions, the grip, the concentricity, or both, may suffer.

B. TAILSTOCK AND QUILL

Tailstock is a very common accessory on a CNC lathe. Its main purpose is to support a part that is too long, too large, or needs to be pressed extra firmly against the jaws, for example, in some rough turning operations. A tailstock may also be used to support a finishing operation of a thin tubular stock, or to support a part that has a shallow grip in the jaws, to prevent it from flying out.

On the negative side, tailstock is usually in the way of tool motions, so make sure to avoid a collision. Atypical tailstock has three main parts:

Tailstock body

Quill

Center

All parts are important in programming and setup.

Tailstock Body

Tailstock body is the heaviest part of lathe tailstock. It is mounted to the lathe bed, either manually during setup, or through a programmable option, hydraulically. Truly programmable tailstock is normally available only as a factory installed option and must be ordered at the time of machine purchase.

Quill

Quill is the shiny cylinder that moves in and out of the tailstock body. It has a fixed range of travel, for example, a 75 mm (3 inch) travel may be found on medium size lathes. When tailstock body is mounted to the lathe bed in a fixed position, quill is moved *out* to support the part, or *in*, to allow a part change. Part itself is supported by a center, mounted in the quill.

Center

Center is a device that is placed into the quill with a tapered end, held by a matching internal taper and is physically in contact with the part. Depending on design, if the tailstock has an internal bearing, a dead center can be used. If tailstock has no internal bearing, a live center must be used instead. Machined part has to be pre-centered (on the CNC lathe or before), using the same tool angle as the tailstock center - normally 60. A typical tailstock is illustrated in *Figure 44-4*.

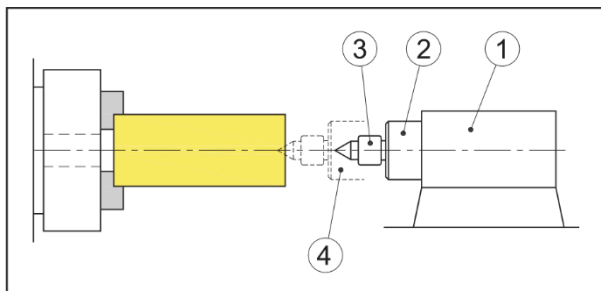


Figure 44-4

Typical tailstock for a CNC lathe:

- (1) Tailstock body (2) Quill - OUT (retracted for work change)
(3) Center (4) Quill - IN (in work support position)

Quill Functions

Programming tailstock quill motion only is just about the same for majority of CNC lathes. There are two miscellaneous functions that work the same way for a programmable and non-programmable tailstock body. These two typical functions are:

M12	Tailstock quill IN or ON = active
M13	Tailstock quill OUT or OFF = active

If quill is supporting the part, it is *in*, using the M12 function. If quill is not supporting the part, it is *out*, using the M13 function. For setup, the M12 and M13 functions may be used, and on many lathes, a toggle switch on the control is provided to operate the quill.

Spindle should be ON when the quill fully supports the part

Programmable Tailstock

Tailstock body is normally not programmable (only the quill is), but this feature is available for many CNC lathes as a *factory installed option*. That means it has to be ordered when making the initial purchase; dealer cannot adapt such option to the machine at a later date. Many different types of programmable tailstocks are available, for example, a slide-type that moves left and right only, or a swing out type, that is out of the way when not needed.

A typical tailstock defined as *programmable* can be programmed using two *non-standard* M-functions. For the example, a CNC lathe may use these two M functions:

M21	Body of tailstock forward
M22	Body of tailstock backward

On some CNC lathes, there may also be two additional M functions available, one of them for *clamping* the tailstock, another for *unclamping* it. In many cases, the two tailstock functions have both clamp/unclamp functions built in.

Atypical programming procedure is to move tailstock towards the part, do some machining and move it back. Instead of presenting an actual programming example, let this procedure serve as a guide - fill-in any M-functions required for a particular CNC lathe:

Unclamp tailstock body

Move tailstock body forward

Clamp tailstock body

Move quill forward into the part

... do the required machining operations ...

Move quill backward from the part

Unclamp tailstock body

Move tailstock backward

Clamp tailstock body

Some procedures take certain amount of time to complete, even if the time is measured in seconds. It is generally recommended to program a dwell function to guarantee completion of one step, before the next step starts. A review of *Chapter 24* may help - see page 175.

Safety Concerns

When programming a job that uses tailstock, safety is at least as important as for other operations. Tool motions programmed towards the part at start, and tool motions returning to tool change position later are both critical. The safest is an approach from tool change position towards the part along Z-axis first, *then* along X-axis. On return from a clear position close to work, reverse the order - first retract the X-axis above the part, *then* move Z-axis (both axes usually move to a safe tool change position).

C. BI-DIRECTIONAL TURRET INDEXING

Another efficiency feature is a *bi-directional turret indexing*. Many CNC lathes have a so called *bi-directional indexing* built-in, that means an automatic method of turret indexing (control decides the direction). However, there is a certain benefit in having a *programmable* indexing direction. If that feature is available on a CNC lathe, there will be two miscellaneous functions available to program turret indexing. Both functions are *non-standard*, so check the machine tool manual, as always in such cases.

Typical M-functions for turret indexing are:

M17	Indexing forward:	T01-T02-T03 ...
M18	Indexing backward:	... T03-T02-T01

Figure 44-5 shows an example of M17 and M18 functions for an eight-sided turret.

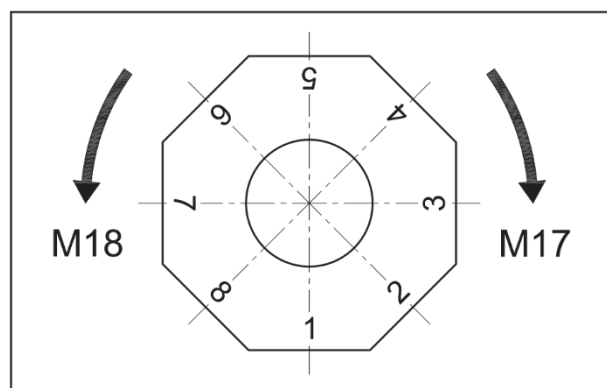


Figure 44-5

Programmable bi-directional turret indexing

In an example, programmer is working with a lathe that has an eight station turret. Tool T01 will be used first, then tool T08 and then back to tool T01 again. There is no problem to index from T01 to T08 *or* from T08 to T01, using the *automatic* turret indexing direction. It makes sense, that a bi-directional turret indexing should be used for efficiency. After all, T01 and T08 may be far apart in numbers but they are next to each other on a polygonal turret with eight stations. Control system will *always choose the shortest method*, in this case, from T01 to T08 in backward direction, then from T08 to T01 in forward direction.

If automatic bi-directional indexing is not built in the machine, it has to be programmed, assuming the control allows that. Otherwise, in normal programming, when going from T08 to T01, indexing motion will *pass all other six stations*, which is rather an inefficient method. The next example shows *how* and *where* to place the M-functions.

Programming Example

This example is a complete program incorporating the bidirectional indexing and also shows how to use a fully programmable tailstock. All tool motions are realistic but not important for the example. *Order of numbering the tools in the turret may not be consistent* from one machine to another! Terms *forward* and *backward* are related to such order. M-functions described earlier are used here:

O4401 (BI-DIRECTIONAL INDEXING AND TAILSTOCK)

N1 G21 G99 M18 (SET INDEX BACKWARD) N2 G50 S1200 (LIMIT MAX RPM)

N3 T0100 (SHORT FROM T02 TO T01 WITH M18)

N4 G96 S500 M03

N5 G00 G41 X98.0 Z5.0 T0101 M08

N6 G01 Z0 F0.75

N7 X-1.8 F0.18 (FACE CUT)

N8 G00 Z5.0

N9 G40 X250.0 Z150.0 T0100

N10 M01

N11 T0800 (SHORT FROM T01 TO T08 WITH M18)
 N12 G97 S850 M03
 N13 G00 X0 Z6.5 T0808 M08
 N14 G01 Z-9.7 F0.125 (#5 CENTER DRILL)
 N15 G04 U0.3
 N16 G00 Z6.5
 N17 X300.0 Z75.0 T0800
 N18 M05 (SPINDLE STOP FOR TAILSTOCK)
 N19 M01 (OPTIONAL STOP)
 N20 M21 (TAILSTOCK
FORWARD)
 N21 G04 U2.0 (2 SECOND DWELL)
 N22 M12 (QUILL IN)
 N23 G04 U1.0 (1 SECOND DWELL)
 N24 G50 M17 (NO MAX RPM - SET INDEX FORWARD)
 N25 T0100 (SHORT FROM T08 TO T01 WITH M17)
 N26 G96 S500 M03
 N27 G00 G42 X79.5 Z2.5 T0101 M08
 N28 G01 X87.0 Z-1.25 F0.2 (ROUGH CHAMFER)
 N29 Z-65.0 F0.3 (ROUGH TURN)
 N30 U0.2
 N31 G00 G40 X250.0 Z150.0 T0100
 N32 M01 (OPTIONAL STOP)
 N33 T0200 (SHORT FROM T01 TO T02 WITH M17)
 N34 G96 S600 M03
 N35 G00 G42 X77.5 Z2.5 T0202 M08
 N36 G01 X85.0 Z-1.25 F0.1 (FINISH CHAMFER)
 N37 Z-65.0 F0.15 (FINISH TURN)
 N38 U0.2 F0.4
 N39 G00 G40 X300.0 Z150.0 T0200
 N40 M05 (SPINDLE STOP FOR TAILSTOCK)
 N41 M01 (OPTIONAL STOP)
 N42 M13 (QUILL OUT)
 N43 G04 U1.0 (1 SECOND DWELL)
 N44 M22 (TAILSTOCK
BACKWARD)
 N45 G04 U2.0 (2 SECOND DWELL)

N46 M30 % (END OF PROGRAM)

This example first uses T01 to face stock to the spindle center line. Then T08 comes in - center drill #5 - and makes a center hole. When the center drill moves into a clear position, tailstock body moves forward and locks, then quill moves into the work. T01 comes back to rough out the chamfer and diameter, after which T02 comes to finish them. When finishing operation is complete, spindle stops, quill moves out, then the tailstock body moves backward. CNC operator always sets the tailstock initial position.

At the job end, T02 is in active position. That means M18 has to be programmed at the program *beginning*, to get a short indexing from T02 to T01 - **a very important step!**

Note how M17/M18 functions are programmed - their location in a particular block is very important. Either function programmed by itself will not cause turret to index - it only *sets* its direction! *Txx00 will make the actual indexing.*

All this leads to one question - can we find out if a particular CNC lathe has built-in *automatic* indexing direction (shortest direction) or *programmable* direction? A good chance is that on CNC lathes where only the *forward* direction takes place (no automatic indexing), there is also a feature called *programmable direction*, available in the form of M17 and M18 miscellaneous - or similar - functions.

Although the tendency on modern CNC lathes is to incorporate automatic turret indexing direction into the control system (which means the *control system is in charge*), there could be major benefits in having the programmable method available for special machining occasions. As an example, think of an oversize tool mounted in the turret. Such tool is perfectly safe to use, as long as it *does not* index the full swing of turret. Automatic indexing has *no provision* for such a situation!

On the other hand, with programmable indexing, CNC programmer has complete control. Programming a setup in a way that will never cause the turret to index full 360° at any time is possible. This may not always be a typical situation - it will take a few seconds extra time, but it can happen quite often for certain type of jobs.

D. BARFEEDER ATTACHMENT

Bar feeder is an external attachment to a CNC lathe that allows small and medium cylindrical parts to be machined without interruption, up to a specified number that can be machined from a single bar of certain length. There are many advantages of using bar feeders, particularly those of modern hydrodynamic design type, rather than the old mechanical design. For example, sawing operations are eliminated (replaced with a much more precise part-off tool), no soft jaws to bore, unattended operation is possible (at least for an extended period of time), stock material economy and high spindle speeds can be achieved on many models with many other advantages.

Bars of material are stored in a special tube that guides the bar (by *pushing* it or *pulling* it) from the tube to the machining area. Two limitations are bar length and bar diameter. They are always specified by the bar feeder manufacturer and as spindle bore diameter of CNC lathe.

Many ingenious designs of bar feeders do exist nowadays, and the programming method is heavily influenced by their design and features.

Functions controlling the chuck opening and closing, block skip function, M99 miscellaneous function and several other special functions, are typical features available for programming bar feeders. Many of these functions had been discussed earlier.

Bar Stopper

Although a bar movement from the guide tube is controlled by the *chuck open* and *chuck close* functions (typically M10 and M11), the bar target position still has to be provided, in terms of how far it has to move out of the guide tube. This position should be lower than the bar diameter and

on the positive side of Z axis (0.025 shown). This is the amount to be faced off (Z0 at the front face assumed). *Figure 44-6* shows such example.

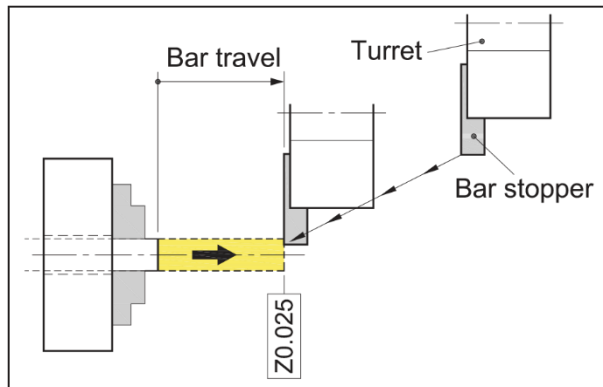


Figure 44-6

Bar stopper position for bar travel

Program itself is quite simple. It will use both M10 and M11 functions, but also another two functions that *may or may not be required* for a particular bar feeder. These *nonstandard* miscellaneous functions are (in the example):

M71	Bar feeder ON - start
M72	Bar feeder OFF - stop

These functions are only examples and may be different for a certain bar feeding mechanism or unnecessary altogether. Here is the sample program with explanations:

O4402

N1 G21 T0100 M05 (T01 IS BAR STOPPER)

N2 G00 X3.0 Z0.6 T0101 (STOP POSITION)

N3 M10 (CHUCK OPEN)

N4 G04 U1.0 (1 SECOND DWELL)

N5 M71 (BARFEEDER ON)

N6 G04 U2.0 (2 SECOND DWELL)

N7 M11 (CHUCK CLOSE)

N8 G04 U1.0 (1 SECOND DWELL)

N9 M72 (BARFEEDER OFF)

N10 G00 X250.0 Z125.0 T0100 (CLEAR POSITION)

N11 M01 (OPTIONAL STOP)

A few important notes relating to a bar stopper may be helpful to develop a better part program:

Tool station #1 (T01) holds the bar stopper (N1)

Initially, chucking of the bar (for each first piece from the bar) is done manually

Spindle rotation must be stopped prior to chuck opening

All miscellaneous functions related to bar feeding should be programmed as separate blocks

Dwell should be sufficient for the task but not excessive

These are some general considerations for programming a bar stopper, but always check the recommended procedures for a particular bar feeder design.

E. ADDITIONAL OPTIONS

There are many other options (non-standard features) on CNC lathes that will qualify as programmable accessories. Some maybe be rather unusual, such as a programmable *chip conveyer*, or programmable *tailstock pressure*, others may not be that rare, for example, a programmable *follower rest* (a moving version of a steady rest), used for ahead-of tool support for long parts. Steady-rest and a follower-rest help prevent chatter or deflection on a relatively long part or a part with thin walls.

Another two common accessories that are also often related to each other - *and to bar feeding as well* - are:

Part Catcher *also known as* **Part Unloader**

Pull-Out Finger

Both are commonly used together with bar feeding operations and use two miscellaneous functions.

Part Catcher or Part Unloader

A very common accessory for continuous machining, using a bar feeder, is *part catcher* or *part unloader*, as it is sometimes called. Its purpose is to catch the completed part after it had been parted-off. Instead of letting the completed part fall into the machine area and possibly cause damage, this attachment will safely intercept each part and move it into a receiving box. Receiving box is often in the area of CNC lathe where machine operator can reach without danger, and without having coolant in the way. There are two *non-standard* miscellaneous functions for a part catcher:

M73	Part catcher advance	... <i>in</i> or <i>forward</i>
M74	Part catcher retract	... <i>out</i> or <i>backward</i>

Following program example illustrates how each function is programmed for a part-off tool.

O4403

```

N1 G21 ...           (TOP           OF
                     PROGRAM)

N81 T0700           (PART-OFF TOOL
                     CTIVE)

N82 G50 S1500       (LIMIT
                     AXIMUM RPM)

N83 G96 S125 M03     (SPINDLE SPEED)

N84 G00 X53.0 Z-67.0 T0707 M08 (START POS.)

N85 M73           (PART CATCHER ADVANCE)

N86 G01 X-0.25 F0.1   (PART-OFF CUTTING)

N87 G00 X53.0 M09 (MOVE ABOVE STOCK DIA)

N88 X250.0 Z125.0 T0700 (SAFE XZ POSITION)

N89 M74           (PART CATCHER RETRACT)

N90 M01           (OPTIONAL STOP)

```

/ N91 M30 (CONTROLLED END OF PROGRAM)

N92 M99 % (RESTART FROM THE TOP OF PROGRAM)

T07 in the program is a 3 mm wide part-off tool, parting off a 50.0 stock diameter to 64 mm length, using standard simple process. Program shows special programming technique, relating to continuous operation. Concentrate on the last three blocks, N90, N91 and N92.

Continuous Operation

Block N90 is an optional stop, typically used for setup and random checking. Block N91 contains M30 - end of program function. Note the slash symbol in front of the block. This is a *block skip* function, described earlier in *Chapter 23*. When block skip switch on the control panel is set to ON position, control system will *not* process the instructions in block N91. That means the program will not end there and processing will continue to block N92, where M99 is programmed.

Although M99 function is mainly defined as the end of a subprogram, it can also be used in the main program - as in this example. In that case, it causes a *continuous* processing loop. M99 function will make the program to return to the top, and - *without interruption* - repeat all processing again. Since the first tool will normally have a bar stopper programmed, bar feeder moves the stock out of the tube and the whole program repeats indefinitely - well, until the block skip switch is set to OFF position. Then, M30 function takes over and M99 in the subsequent block will *not* be processed, as the program has ended.

Parts Counter

This kind of unattended lathe machining often uses another control system feature - *parts counter*. Parts may be counted via a program (usually a user macro), or by setting the number of required parts in control system memory. Counters may also be programmed by *non-standard* miscellaneous functions, for example:

M88	Count up ... <i>ascending order</i>
M89	Count down ... <i>descending order</i>

Preset number for the count is usually the bar capacity or the required number of parts from a single bar. Programming example at the end of this chapter will illustrate the counter function and some other features.

Pull-Out Finger

As the name suggests, a pull-out finger is a device (CNC lathe accessory) that *grabs and pulls* the bar out of bar feeding guide tube (while the chuck is open). This is a typical method for bar feeders of the '*pull-type*'. Normally, pull-out finger is mounted in the turret, either as a separate 'tool', or as an add-on to an existing tool, in order to preserve the number of available tool stations. Since these activities cannot be used with spindle rotating, yet they often need a feed rate, they are programmed in the G98 mode *feed rate per time* (mm/min or in/min).

Regardless of the exact pull-out finger model available, programming procedure is just about the same - no bar extends from the spindle longer than its face after part-off:

At a safe start position, index to the tool station where pull-out finger is mounted. Spindle must be stopped at this time with M05!

At a rapid rate, move to spindle centerline (X0), and a Z axis position about half-way of the overall bar projection

In 'feed-per-time' mode, feed-in towards the bar as extended after part-off

Dwell for about 0.5 second for the finger to catch the bar stock

Open the chuck with M10

Pull out bar stock from the guide tube

Dwell for about 0.5 second for the finger to complete the pull-out

Close the chuck with M11

Dwell for about 1 second to complete chuck closing

Move pull-out finger away from the bar stock

Return pull-out finger to the safe start position

Reinstate the 'feed-per-revolution' mode

In programming terms, the general structure will be similar to this format (item numbers correspond the list):

O4404 ...

N.. ...

N.. Txx.. M05 (ITEM 01)

N.. G00 X0 Z.. (ITEM 02)

N.. G98 G01 Z.. F.. (ITEM 03)

N.. G04 U0.5 (ITEM 04)

N.. M10 (ITEM 05)

N.. G01 Z.. F.. (ITEM 06)

N.. G04 U0.5 (ITEM 07)

N.. M11 (ITEM 08)

N.. G04 U1.0 (ITEM 09)

N.. G00 Z.. (ITEM 10)

N.. X.. Z.. Txx00 (ITEM 11)

N.. G99 (ITEM 12)

N.. ...

Feel free to modify the program structure to suit the requirements of any unique setup in the machine shop.

F. PROGRAMMING EXAMPLE

The following programming example illustrates a complete program for an unattended bar feeding operation, until the number of parts have been machined. CNC lathe operator sets the required number of parts when starting a new bar stock. This program requires a careful study. It does contain some very practical and advanced features, all of them already discussed, mostly in this chapter:

O4405

(N1 TO N18 FOR NEW BAR ONLY - 40 MM CUT-OFF)

N1 M18 (INDEX T03 TO T01)

N2 G21 T0100 M05 (T01 - BAR STOPPER)

N3 G00 X2.5 Z40.0 T0101 (NEW BAR OUT 40
M)

N4 M10 (CHUCK OPEN)
 N5 G04 U1.0 (1 SECOND DWELL)
 N6 M71 (BARFEEDER ON)
 N7 G04 U1.0 (1 SECOND DWELL)
 N8 M11 (CHUCK CLOSE)
 N9 X150.0 Z50.0 T0100 (CLEAR POSITION)
 N10 M01 (OPTIONAL STOP)
 N11 M17 (INDEX T01 TO T03)
 N12 T0300 (T03 - 3 MM WIDE PART-OFF TOOL)
 N13 G97 S1400 M03 (CUTTING RPM)
 N14 G00 X32.0 Z0 T0303 M08 (START POSITION)
 N15 G01 X-0.5 F0.1 (PART-OFF BAR BAD END)
 N16 G00 X32.0 M09 (MOVE ABOVE BAR)
 N17 X150.0 Z50.0 T0300 (CLEAR POSITION)
 N18 M01 (OPTIONAL STOP)
 N19 M18 (INDEX T03 TO T01)
 N20 T0100 M05 (T01 - BAR STOPPER) N21 G00 X2.5 Z1.0 T0101 (1 MM STOCK ON
 FACE)
 N22 M10 (CHUCK OPEN)
 N23 G04 U1.0 (1 SECOND DWELL)
 N24 M71 (BARFEEDER ON)
 N25 G04 U1.0 (1 SECOND DWELL)
 N26 M11 (CHUCK CLOSE)
 N27 X150.0 Z50.0 T0100 (CLEAR POSITION)
 N28 M01 (OPTIONAL STOP)
 N29 M17 (INDEX T01 TO T02)
 N30 T0200 (T02 - FACE-CHAMFER-TURN OD)
 N31 G96 S120 M03 (CUTTING SPEED)
 N32 G00 G41 X32.0 Z0 T0202 M08 (START FACE)
 N33 G01 X-1.8 F0.175 (FACE-OFF FRONT)
 N34 G00 Z2.5 (CLEAR Z+)
 N35 G42 X17.5 (CHAMFER START)
 N36 G01 X24.0 Z-0.75 F0.08 (CUT CHAMFER)
 N37 Z-29.0 F0.12 (CUT DIAMETER TO LENGTH)
 N38 U5.0 F0.2 (CLEAR ABOVE BAR)
 N39 G00 G40 X150.0 Z50.0 T0200 (CLEAR POSITION) N40 M01 (OPTIONAL
 STOP) N41 T0300 (T03 - 3 MM WIDE PART-OFF TOOL)
 N42 G97 S1400 M03 (CUTTING RPM)

N43 G00 X32.0 Z-28.0 T0303 M08 (START POS.)
N44 G01 X-0.5 F0.1 (PART-OFF TO 25 MM LENGTH)
N45 G00 X32.0 (MOVE ABOVE BAR)
N46 X150.0 Z50.0 T0300 (CLEAR POSITION)
N47 M01 (OPTIONAL STOP)
N48 M89 (UPDATE PART
NUMBER BY 1)
/ N49 M30 (CONTROLLED END
PROGRAM)
N50 M99 P19 % (RESTART FROM
BLOCK N19)

As it usually goes with accessories and options, the machine tool manufacturers use a number of M-functions to activate and deactivate a particular accessory. It is not possible to cover any specific procedures into a general reference material. Hopefully, ideas presented in this chapter will help to adapt to any manufacturer's recommendations and understand them better.

34. WRITING A CNC PROGRAM

Writing a CNC program is the final result of manual programming. This last step requires a sheet of paper, or many sheets of paper, that contain the part program. Program itself is composed of individual instructions related to machining and arranged in a series of sequential blocks. Writing does not mean using only a pen or pencil, on the contrary. Instead of paper, modern writing methods employ a computer and a text editor, but the result is still a written and or printed copy of a manually generated part program.

Manual program development is the result of a lot of hard work. A short program with a few lines of code may be as easily entered into the control directly as to be written down on paper. However, the written copy will often be required for documentation and other reference purposes.

The need to program by hand seems somewhat backwards in the age of computers, printers and other hi-tech wonders, but it is a method that will not disappear any time soon, at least not for simple programs. Writing a part program manually requires time and is always subject to errors. Manual work means work by hands, so it seems that a need for special computer skills is not required. Is that a correct assessment?

In the traditional way, program can be written with a pencil and a paper (and a five pound eraser, as an old cartoon claimed). Its final form is transferred to the control unit, a short program may be keyed into the system directly, by pressing various keyboard keys. For long programs, this approach is a waste of time. Modern alternative to a pencil is a computer keyboard, using a simple text editor to make a plain ASCII text file, with no formatting. Computer software creates a CNC program as a file stored on the hard drive. This file can be printed or send directly to CNC machine. The only difference is that computer keyboard has replaced the pencil and editing features of a text editor have replaced the eraser. Even today, a large amount of manual programming work in is still done in writing, using a device such as pens, pencils, calculators and erasers.

Regardless of media used, learn how the computer - the control system - interprets a written program, what syntax to use, what to avoid and what format is correct. Even if not programming manually at all, it is important to know the principles of program writing techniques, in order to make changes or customize output that was developed by a CAD/CAM system, if necessary.

CNC program should be written in such a way that it can be interpreted without a difficulty

A. PROGRAM WRITING

Writing all collected data into a final version of CNC part program is one of the last items of programming process. To get to this stage requires hard work through all other stages - but only when all thoughts have been collected, all decisions have been made and a certain level of comfort has settled in. In the previous chapters, emphasis was on program development as a logical process. Now, the focus will shift to the actual method of writing CNC program, following this logical process.

Writing a part program is based on two initial factors:

Corporate standards ... **company decides** Personal **style** ... **you decide**

Both factors can be adapted simultaneously in a single program - they are fully compatible. It is unreasonable to expect any *industry or world-wide standards* relating to various techniques of developing a program. It may be even less reasonable to set any *company based standards*, unless there is a general set of rules and recommendations already in existence.

The final result is that the first guiding factor - company standards - is replaced by the second factor - personal style. From an objective point of view, there is nothing wrong at all with a personal style of programming. If the program works, who cares how it was done. From a revised point of view, it needs to be acknowledged that a CNC programmer can never succeed in isolation. Programming involves at least one user of the final program - CNC operator - and that makes

programming, in effect, a true teamwork.

The most common problem with uncontrolled personal style is *inconsistency*. Any CNC machine shop that employs - or plans to employ - more than one programmer, should establish certain minimum standards for preparation of a part program. Adherence to these standards allows any team member to pick up where another member has left. Often, personal style of the first programmer in the company will carry on and on and eventually becomes 'the' company standard, for better or worse. Such a situation may well be very positive, but in most cases it needs revaluation or at least a bit of modernizing.

To define a company standard, first evaluate some suggestions and practical observations that may be helpful to prepare a program efficiently for any style that may be suitable to follow and useful in the future.

Legibility of Handwriting

Writing a program without any assistance of a computer and a text editor, means writing a CNC program in pen or pencil, by hand. A handwritten program is easier to correct without a mess and it should be double or even triple spaced when written on a sheet of paper. Individual words in a program block should be separated by a space, to further enhance legibility. This way, any additions or future changes (if they become necessary) can be made quite easily, yet still keeping the overall appearance of paper copy clean and neat. Problems with legibility of a manually generated program are much less of a factor, if the program is typed directly into a computer text file or using an CNC editor, such as the one included with NCPlot software. Even in those cases, the printed copy may be illegible for technical reasons, such as a low printer toner, for example.

Programming Forms

In the early years of numerical control, special programming forms were popular with pre-printed columns for each address in a block. Those were the days when only numerical values were entered into the appropriate column and the column position determined the address meaning. These programming forms were often issued by control and machine manufacturers, as an aid to program writing and a little promotion on a side. Today, a ruled sheet of a standard size paper is fully sufficient. No special columns are needed and if a column or two is justified, it can be drawn easily enough. Modern programs use alphanumeric representation, writing the whole word - alpha characters as well as numeric characters and special symbols. This process is much more economical - hardly any machine manufacturers provide programming forms anymore.

Some CNC programmers do not necessarily do the final program version themselves. Some shops consider such work a secretarial responsibility. That means somebody else (an assistant) will read the handwritten copy and has to be able to read it correctly, the way it was intended. Such a person may have no knowledge of CNC programming and may not be able to detect even simple syntax errors.

Confusing Characters

Legibility of programmer's handwriting is very important. Make a special effort when writing certain characters (alphabetical or numeric) that can be interpreted in more than one way. Depending on personal handwriting, some characters can be confusing to the reader. For example, a handwritten letter O and digit 0 can look the same. Digit 2 and the letter Z can be also confusing. The letter I and the digit 1 as well as a low case letter *l* are other examples. These are only some of the most obvious examples, but many other characters can also be confusing, depending on each person's handwriting. Try to develop a consistent writing technique to distinguish potentially confusing characters (confusing is a relative term, of course).

For instance, all computers and printers (even the old tape preparation systems) use a special type of style to distinguish individual characters on screen and in print. In this digital book there is an obvious difference between a wide letter **O** (as in **O1111**) and a narrow digit **0** (as in **00001**).

The same technique should be applied to personal handwriting. Take advantage of the fact that there is no letter O used on most controls anywhere, except as a program number and in a comment section, where a misprint will not create a problem anyway. If preferred, find a special designation only for the letter O and the rest is all digits 0 by default - unless hundreds of zeros are identified specifically and in a unique way in every part program.

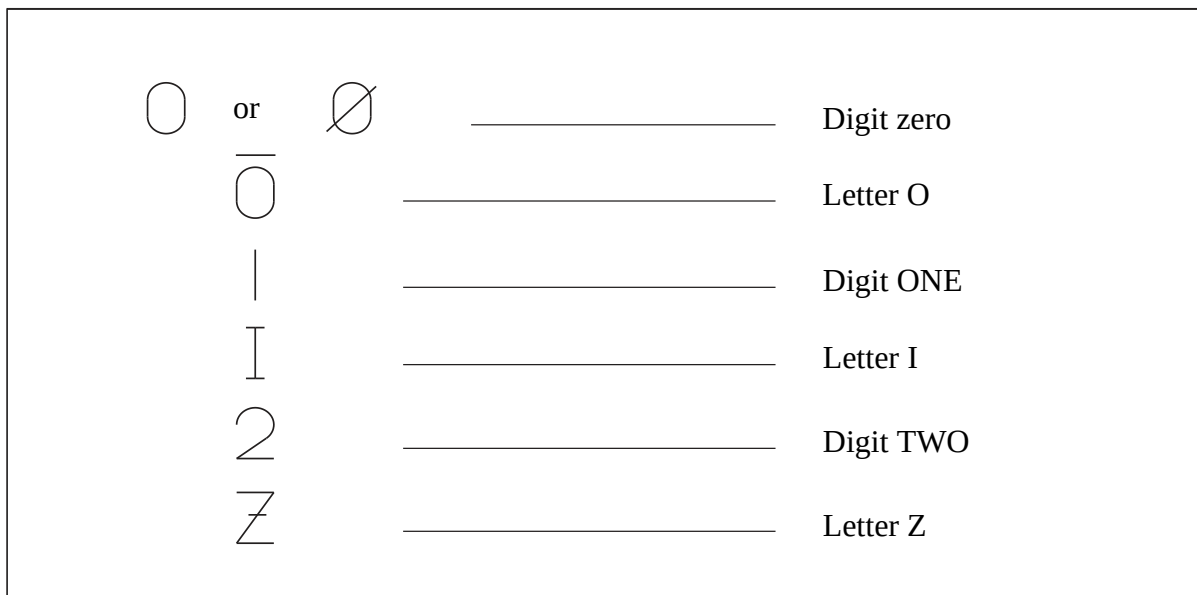


Figure 48-1

Special form of characters written to prevent ambiguity

Illustration in Figure 48-1 shows some *suggested* methods of common character distinction in handwriting. Find a personal way to write any characters that may improve handwriting legibility. Regardless of which method is selected as personally preferable, adhere to it - *be consistent*. There is nothing worse than adopting different 'standards' for every new program. Programmer or the person who prepares the program final version, will be more than confused and eventually may make a serious error.

Handwritten method can be bypassed entirely by keying in the program data via control keyboard. Such program can then be optimized, part machined and verified program sent out when the job is finished. This procedure may tie up the machine for a while and is not recommended as an everyday method of inputting programs into the control system. The best and fastest method is to prepare part program in a text editor on a computer and send it directly to the CNC machine, through cable connection or memory card.

Only a small minority of CNC users today use punched tape, and even then, it is usually for old machines only. More modern methods are available, such as disk storage of a desktop or laptop computer. Through an interface between computer and the machine, data can be transferred either way very reliably, thus eliminating punched tape and other methods altogether. Whichever method is selected as preferable, the program still has to be properly formatted.

WRITING A CNC PROGRAM 469

B. PROGRAM OUTPUT FORMATTING

Those who followed this digital book from its beginning, chapter by chapter, should be well familiar with programming by now. This section deals with the actual program format - not *its contents*, but how it *appears* on the printed paper or screen of the computer. It will evaluate four versions of the *same* program. Identical in every respect, except its outward appearance. Feel free to be the judge as to which of the four format versions is the most suitable and why. A rather long program is presented - intentionally this is an actual program. It is not important what it does, only

how it looks when printed or displayed. Each new version presented improves the previous version.

Program *Version 1* - Imperial units used:

G20

G17G40G80G49

T01M06

G90G54G00X-32500Y0S900M03T02

G43Z10000H01M08

G99G82X-32500Y0R1000Z-3900P0500F80

X32500Y32500

X0

X-32500

Y0

Y-32500

X0

X32500

G80G00Z10000M09

G28Z10000M05

M01

T02M06

G90G54G00X-32500Y0S750M03T03

G43Z10000H02M08

G99G81X-32500Y0R1000Z-22563F120

X32500Y32500

X0

X-32500

Y0

Y-32500

X0

X32500

G80G00Z10000M09

G28Z10000M05

M01

T03M06

G90G54G00X-32500Y0S600M03T01

G43Z10000H03M08

G99G84X-32500Y0R5000Z-13000F375

X32500Y32500

X0

X-32500

Y0

Y-32500

X0

X32500

G80G00Z10000M09

G28X32500Y-32500Z10000M05

M30 %

This is the most primitive way of writing a program. Although it may offer some doubtful benefits, it is the least friendly version. Good program, yes, but with a very poor appearance. It is extremely difficult for CNC operator to read this mess.

Program *Version 2* - Imperial units used:

N1G20

N2G17G40G80G49

N3T01M06

N4G90G54G00X-3.25Y0S900M03T02

N5G43Z1.0H01M08

N6G99G82X-3.25Y0R0.1Z-0.39P0500F8.0

N7X3.25Y3.25

N8X0

N9X-3.25

N10Y0

N11Y-3.25

N12X0

N13X3.25

N14G80G00Z1.0M09

N15G28Z1.0M05

N16M01

N17T02M06

N18G90G54G00X-3.25Y0S750M03T03

N19G43Z1.0H02M08

N20G99G81X-3.25Y0R0.1Z-2.2563F12.0

N21X3.25Y3.25

N22X0

N23X-3.25

N24Y0

N25Y-3.25
N26X0
N27X3.25
N28G80G00Z1.0M09
N29G28Z1.0M05
N30M01
N31T03M06
N32G90G54G00X-3.25Y0S600M03T01
N33G43Z1.0H03M08
N34G99G84X-3.25Y0R0.5Z-1.3F37.5
N35X3.25Y3.25
N36X0
N37X-3.25
N38Y0
N39Y-3.25
N40X0
N41X3.25
N42G80G00Z1.0M09
N43G28X3.25Y-3.25Z1.0M05
N44M30
%

This is definitely a much improved version of the same program. Look at what a simple block numbering and inclusion of decimal points does to overall program legibility. This program version is far from being final, but it does offer some very tangible improvement. Decimal points in programs are of course standard, except for old controls.

The next program version applies all improvements done so far and addresses some additional issues.

Program Version 3 - Imperial units used:

N1 G20
N2 G17 G40 G80 G49
N3 T01 M06
N4 G90 G54 G00 X-3.25 Y0 S900 M03 T02
N5 G43 Z1.0 H01 M08
N6 G99 G82 X-3.25 Y0 R0.1 Z-0.39
N7 X3.25 Y3.25
N8 X0
N9 X-3.25
N10 Y0

N11 Y-3.25
N12 X0
N13 X3.25
N14 G80 G00 Z1.0 M09
N15 G28 Z1.0 M05
N16 M01
N17 T02 M06
N18 G90 G54 G00 X-3.25 Y0 S750 M03 T03
N19 G43 Z1.0 H02 M08
N20 G99 G81 X-3.25 Y0 R0.1 Z-2.2563 F12.0
N21 X3.25 Y3.25
N22 X0
N23 X-3.25
N24 Y0
N25 Y-3.25
N26 X0
N27 X3.25
N28 G80 G00 Z1.0 M09
N29 G28 Z1.0 M05
N30 M01
N31 T03 M06
N32 G90 G54 G00 X-3.25 Y0 S600 M03 T01
N33 G43 Z1.0 H03 M08
N34 G99 G84 X-3.25 Y0 R0.5 Z-1.3 F37.5
N35 X3.25 Y3.25
N36 X0
N37 X-3.25
N38 Y0
N39 Y-3.25
N40 X0
N41 X3.25
N42 G80 G00 Z1.0 M09
N43 G28 X3.25 Y-3.25 Z1.0 M05
N44 M30
%

This version is much improved. It uses all improvements of the previous version yet adds a significant improvement - *spaces between words*. Still, it is difficult to visually identify the start of each tool. The next version will add a blank line between tools. Spaces do not impose an extra

drain on CNC memory, yet the program is much easier to read.

Program Version 4 - Imperial units used:

(DRILL-04.NC)

(PETER SMID - 07-DEC-08 - 19:43)

(T01 - 1.0 DIA - 90DEG SPOT DRILL)

(T02 - 11/16 TAP DRILL - THROUGH) (T03 - 3/4-16 TPI PLUG TAP)

(T01 - 1.0 DIA - 90DEG SPOT DRILL)

N1 G20

N2 G17 G40 G80 G49

N3 T01 M06

N4 G90 G54 G00 X-3.25 Y0 S900 M03 T02

N5 G43 Z1.0 H01 M08 (INITIAL LEVEL)

N6 G99 G82 X-3.25 Y0 R0.1 Z-0.39 P0500 F8.0

(HOLE 1)

N7 X3.25 Y3.25 (HOLE 2)

N8 X0 (HOLE 3)

N9 X-3.25 (HOLE 4)

N10 Y0 (HOLE 5)

N11 Y-3.25 (HOLE 6)

N12 X0 (HOLE 7)

N13 X3.25 (HOLE 8)

N14 G80 G00 Z1.0 M09

N15 G28 Z1.0 M05

N16 M01

(T02 - 11/16 TAP DRILL - THROUGH)

N17 T02 M06

N18 G90 G54 G00 X-3.25 Y0 S750 M03 T03

N19 G43 Z1.0 H02 M08

N20 G99 G81 X-3.25 Y0 R0.1 Z-2.2563 F12.0

(HOLE 1)

N21 X3.25 Y3.25 (HOLE 2)

N22 X0 (HOLE 3)

N23 X-3.25 (HOLE 4)

N24 Y0 (HOLE 5)

N25 Y-3.25 (HOLE 6)

N26 X0 (HOLE 7)

N27 X3.25 (HOLE 8)

N28 G80 G00 Z1.0 M09
N29 G28 Z1.0 M05 N30 M01
(T03 - ¾-16 PLUG TAP)
N31 T03 M06
N32 G90 G54 G00 X-3.25 Y0 S600 M03 T01
N33 G43 Z1.0 H03 M08
N34 G99 G84 X-3.25 Y0 R0.5 Z-1.3 F37.5(HOLE 1)
N35 X3.25 Y3.25 (HOLE

N36 X0 (HOLE

N37 X-3.25 (HOLE

N38 Y0 (HOLE

N39 Y-3.25 (HOLE

N40 X0 (HOLE

N41 X3.25 (HOLE
N42 G80 G00 Z1.0 M09
N43 G28 X3.25 Y-3.25 Z1.0 M05
N44 M30
%

The final version (*Version 4*) may be a luxury for some users, but it is the most elegant of all four. It adds initial descriptions and messages to the operator. It includes programmer's name and date of the last update. It also includes description of all tools at the program beginning. It uses the same tool descriptions for individual tools, at the beginning of each tool section, where it matters most.

Some lower level controls do not accept comments in the program. If there are comments in the program, such a control system will strip them automatically during loading.

C. LONG PROGRAMS

Those who ever worked with a punched tape, used to run the program directly in a reel-to-reel operation. The maximum program length equaled the maximum length of tape that fitted on the reel, about 900 feet - or 275 meters - or 108000 characters. With today's modern equipment, there is no need for a tape anymore, most part programs will run from memory of the CNC system. Unfortunately, even that memory capacity is finite as well, often well below what the tape capacity used to be. It all means that a situation may arise, when a particularly long program will not fit into the memory. In addition to a good directory cleanup, there are two other possibilities to eliminate this problem.

Program Length Reduction

The simplest way to reduce a program length is to eliminate all *unnecessary* characters from

such program. Since the problem is related to a *long program*, the reduction in length will be much greater than can be illustrated here. There is a number of areas that should be considered before taking the proverbial red pen and starting the changes:

Eliminate all unnecessary leading or trailing zeros

(G00 = G0, X0.0100 = X.01, ...)

Eliminate all zeros programmed for convenience (ex.: X2.0 = X2.)

Eliminate all or most block numbers

If using block numbers, increments by one will make a shorter program

Join several single tool motions into a multi-axis tool motion, if safety allows

Use default control settings, but check them first

Do not include program comments and messages to the CNC operator

Use comments and various descriptions on a separate piece of documentation

Organizing the programming process will definitely help - for example, include as many instructions in a single block as possible, rather than dividing them into many individual blocks. Use subprograms if possible, use fewer tool changes, even fewer tools, if that is possible, etc. At the same time, watch for undesirable side effects when eliminating or deviating from an established program format.

There is no doubt that many of these measures will result in some compromise between convenience and necessity. When thinking well ahead and organizing the work properly, the results will be worth the effort.

These methods are shortcuts and should be used for emergency situations only, not as standard programming procedures

To illustrate at least some shortcut methods, compare the two following examples - both will have identical results well, *almost* identical results:

O4801 (TYPICAL PROGRAM - METRIC)

N10 G21 G17 G40 G80 G90

N20 G54 G00 X120.0 Y35.0

N30 G43 Z25.0 H01

N40 S500 M03

N50 M08

N60 G99 G81 X120.0 Y35.0 R3.0 Z-10.0 F100.0

N70 X150.0

N80 Y55.0

N90 G80 G00 Z25.0

N100 M09

N110 G28 X150.0 Y55.0 Z25.0

N120 M30

%

A grand total of 194 characters have been programmed. The condensed version of this program needs only 89 characters, with a minor compromise. Program in this form is more memory efficient but much harder to read - remember this is only a short sample, not a real long program in its

entirety, where the difference would be more impressive:

O4802

G90 G0 X120. Y35.

G43 Z25. H1 S500 M3

M8

G99 G81 R3. Z-10. F100.

X150.

Y55.

G80 Z25. M9

G91 G28 X0 Y0 Z0

M30

%

Total of 54.12% of program length have been saved in a rather very short program. Shortening program length may become useful in some cases, so here are several methods that have been used in the above example:

Program description has been eliminated

Block numbers have been eliminated

G21, G17 and G54 have been eliminated

(Correct settings assumed on the control - *be careful!*)

Zeros following a decimal point in full number have been canceled

Some blocks were joined together

G80 G00 has been replaced by G80 only

(G00 is redundant, although commonly used)

Leading zeros in G00, M08, M09, H01, and M03 have been removed

Machine zero return has been changed from absolute mode to incremental mode

... Keep in mind, this is a no-frills program only!

Although both program examples will result in a part machined according to drawing specifications, some programming instructions will be processed differently. A very important change can be achieved in tool approach towards the part. In the first example, usually a standard version, the motion command positions the X and Y axes first, with the Z-axis motion following in a separate block. In the shorter example, the same order of motions has been preserved for safety reasons. If machining conditions allow, these two motions can be combined into a single motion. G43 and G54 commands can work quite well together in the same block, without a problem:

G90 G0 G43 G54 X120. Y35. Z25. H1 S500 M3

Always be careful to consider machine setup first and guarantee a safe approach towards or away from the machined part. If any obstacles come in the way because of any shortcut, the condensed example would be a wrong programming method and should be avoided.

Program preparation and its actual writing will become a daily routine very soon after establishing a personal programming style. If using a computer, learn how to write the program directly from the keyboard - it is a waste of time to write it by hand first. It may take a little time getting used to, but it is well worth it.

Memory Mode and Tape Mode

Most CNC system have a special *Mode Switch* selector to choose from at least two options - MEMORY mode and TAPE mode. *Memory mode* is used most frequently - program is loaded into the CNC memory, it is edited from the memory, and is run from the same memory. Tape mode is, of course, to run a program from a tape and many users ignore the possibilities this mode offers. Even if not using punched tapes in the machine shop anymore, (majority of companies do not), tape mode can be used to *emulate* a tape with many added benefits. Tape mode is not to be taken literally anymore. Think of this mode as an *external* mode (DNC, for example), not in its actual old fashioned sense.

In order to use this external mode requires a little extra hardware and software. On the hardware side, only a reliable personal computer is needed, with a fair size hard disk storage capacity. A properly configured cable that will connect the computer with the CNC is also required. The computer may even be an 'abandoned' slower computer from the office - all that is needed is a minimum configuration, nothing fancy. On the software side, an inexpensive communications software is necessary, to send a program from the computer to the CNC and back.

Once everything is configured to work together, store the CNC program or programs on the hard disk of a personal computer, load the software and work with the CNC system as usually! The major difference is in editing. Since the program actually resides on the hard disk of a PC, use the computer and a text editor to edit a CNC program in that environment, not at the control system. The capacity of current hard drives is *x-times* more than will ever be needed. Aircraft companies, many mold shops, tool and die shops and other industries that require extremely long programs have embraced this technology long time ago, and very successfully, too.

Also consider this method for the *High Speed Machining* programs. This relatively new technology uses very high spindle speeds and feed rates but very small depths of cut. This combination means extremely long programs, many that will not fit into any memory configuration of any CNC system. So before investing into rather expensive control memory updates, investigate this method of running a part program from a personal computer, if the transfer speed is fast enough.

35. PROGRAM DOCUMENTS

During program preparation, quite a number of individual pieces of documentation will accumulate. All sketches, calculations, setup sheets, tooling sheets, job descriptions, instructions to the operator and related notes contain valuable information. This information should be stored as part of the program documentation folder. Any changes to the finished program at a later date, for whatever reason, can be done much easier if the documentation is complete, well organized and stored in one place. A good documentation makes a review of the program at a later date much easier. If somebody else has to review a program, the documentation will save much of the valuable time. The way programmers document programs reflects not only their personal programming style, but it also becomes a reliable indication of their sense of discipline and organizational capabilities.

A simple definition relating to program documentation can be presented:

Program documentation is a set of all records necessary to retrace the program development

Many CNC programmers, even machine shop supervisors, underestimate the importance of good program documentation. Their main arguments are that any paperwork is not worth the time, that it takes too long to collect all documents and prepare the documentation, that it is essentially a nonproductive effort, etc. These arguments are true, up to a point - in order to make a good documentation, yes, some time will be required. Not an excessive amount of time, but enough time to do a good job. If there are prepared blank forms available, they just need to be filled. It does not take any more time than writing the same information on any other piece of paper - it can actually take a lot less time. If a CAD system is available, use it to develop a customized tooling library and setup sheet. A variety of blank forms can be predefined, then filled quickly whenever they are needed. CAD system will save time, it makes the program documentation neat, and every sketch drawn in scale can be easily retraced. Using a word processing or a spreadsheet software is another way to save time required for documentation.

In essence, the purpose of a program documentation is to communicate programmer's ideas to somebody else or to review them at a later date. Creating documentation is not a directly productive work and does not require to take too much extra time. Documentation may be a good investment in time management, it can save a lot of time over a few months.

A. DATA FILES

A complete part program is not only a hard copy of the program or program data (usually stored on a disk). Examples of the important documents mentioned here are all a vital part of the program. They create a set of all files used for programming, called the *data files*.

All of these files are useful to the programmer, but only some are important to the CNC machine operator or the setup person. A number of files are only for reference and are not normally sent to the machine shop. Two basic rules for data files can be established: **Programmer keeps all the files**

Machine operator gets copies of relevant files only

These two rules guarantee that the ultimate responsibility for program control remains with CNC programmer. There is no need to duplicate every piece of documentation for the machine shop - only those items that relate to the actual machining have a place in the shop. Unnecessary duplication is counterproductive and should be avoided. The only items of documentation needed in the machine shop are:

Part drawing

Program printout

Setup sheet

Tooling sheet

Part drawing serves as a reference for comparison of the intended shape, dimensions, tolerances, etc., with the actual product. Only the drawing version that was actually used for programming should be considered. *Program printed copy* is the program listing made available to the machine operator, if required. Normally, it is the printed output of the program itself. Two remaining items, *setup sheet* and *tooling sheet* describe programmer's decisions relating to the part setup and the cutting tools selection. Each is also a complement to the program itself. In some cases, other (special) documentation has to be included as well, not mentioned here. Any piece of paper or a note that is considered important should be included.

Those who have written programs in a high level language (C++, *Visual Basic*, etc.) or in older languages such as *Basic*, *Pascal* or even *Auto LISP* (the original programming language for AutoCAD), know that they can add comments within the body of the program.

These comments are usually terse, just long enough to remind the user of what is happening in the program. If more information about a specific program is necessary, most likely there would be additional instructions, even a user's manual. This kind of external and internal program documentation applied in software development, is also adaptable to a CNC program.

B. PROGRAM DOCUMENTATION

The difference between an external and internal program documentation deserves some explanation. Is one better than the other? Which one should be used?

The best documentation is one that combines both types for maximum effect. To distinguish between the two types, let's evaluate them individually.

External Documentation

External documentation of a CNC program generally consists of several items and always of their latest version this last statement is very important.

The following items are typical to any program documentation. They can be used (or ignored) as desired:

Program copy printout

Methods sheet, if available

Part drawing

Working sketches and calculations

Coordinate sheet

Setup sheet

Tooling sheet

Program data (disk or other media)

Special instructions

Program copy printout is the final version of the programming process. It should be the *exact* contents of the program stored on a disk or other media. In those machine shops that use routing or methods sheets, the programmer should make it a policy to include a copy of any methods sheet in the documentation as well. Part drawing (or its copy) is extremely important to be kept together with the program. It is the ultimate reference source for the future. All sketches and calculations, together with a sheet of coordinates, are also useful at a later date, particularly if the program has to be revised for some reason. Setup sheet and tooling sheet will be discussed shortly.

That only leaves the program data source (usually stored on a disk or similar media) to be included in the documentation folder and any special instructions that may be required by the programmer, CNC machine operator or somebody else.

Internal Documentation

Program internal documentation is contained *within the body of a program* itself. When writing a program, make an effort to strategically place comments into the program. Such messages are integral part of the program and are categorized as *internal program documentation*. These messages are either separate blocks of a program or additions to individual blocks (delimited by parentheses) and can be actually seen on the display screen during program execution (on most controls). They are also printed in a copy of the program. The greatest advantage of internal documentation is the convenience offered to CNC machine operator. The only disadvantage is that when loaded into the control memory, comments do occupy memory space. If the available memory is scarce, be modest with program comments and place more emphasis on external instruction methods. All program comments, messages, directions and instructions must be enclosed in parentheses:

(THIS IS A COMMENT, MESSAGE OR INSTRUCTION)

Parentheses are the required program formatting. Either comment, message or instruction can be an individual block in the program, or it can be part of a program block. Control system will ignore all characters between the parentheses. To avoid long descriptions internally, use pointers to external documentation instead - for example:

N344 ...

N345 M00 (SEE ITEM 4) N346 ...

ITEM 4 in the program comment section will be a detailed description that relates to block N345, somewhere else in the program documentation, such as in a setup sheet. This kind of reference is useful when the message or comment would be too long to be stored in the program body.

For example, CNC operators may find the referenced ITEM 4 in the setup sheet, under a unique heading called *Special Instructions*:

...

ITEM 4. - Remove part, clean jaws, reverse and clamp on the 120 mm diameter

...

Properly prepared internal documentation should always briefly describe each cutting tool used:

N250 T03

N251 M06

(T03 = 1 INCH DIA 4-FLT E/M) N252 ...

Note that T03 is the *current* tool. This designation will vary greatly, depending on tool changing methods of a particular machine tool builder. Also note the use of abbreviations in the program comment - **4-FLTE/M** is a short form for a *4-flute end mill*.

Every time *Program Stop* command M00 is used in the program, document the reason why it is used:

N104 G00 Z1.0

N105 M00 (CHECK DEPTH = 0.157 INCHES)

N106 ...

For longer messages, make the comment a separate block:

N104 G00 Z1.0

N105 M00

(DEPTH TO SHOULDER MUST BE 0.157 INCHES)

N106 ...

Comments can be in the same block as program data:

N12 G00 X3.6 Z1.0

N13 M05 (CLEAN CHIPS FROM THE HOLE) N14 ...

If the comment is written as a separate line, block number for the comment block is usually not used. Instructions that are part of the documentation should be clearly understood. Enigmatic or cryptic messages will not do. Distinct messages translate into time saving by the CNC operator or the setup person and they contribute to a quicker turnaround between individual jobs.

Program Description

On many Fanuc systems, program *description* can also be documented. This is a special kind of a comment, also included in parentheses. There are some conditions that make the program description special.

Description must be included in the same block as program number

Description must have no more than 15 characters

Low case characters will not be accepted

A typical example of a program description may include drawing name and/or number in the comment section:

O4901 (FLANGE-DWG.42541)

Once these conditions are followed, program number can be viewed along with its description right on the directory screen (program listing) of the control system.

If an additional description that does not fit the 15 characters limit is needed, enter more comments in subsequent blocks. They will not be seen on the directory screen but can still be handy for internal program documentation. They will be displayed during program processing on all controls that accept comments. The length of these comments is not usually limited to 15 characters:

O4902 (RING-OP.1)

(DWG. A-8462 REVISION D)

(PETER SMID - 07-DEC-08) N1 ...

The main purpose of program documentation is to transfer all important decisions and ideas from the developer to the user. In CNC programming environment, documentation transfers all ideas from the programmer's desk to the machine shop. It serves as an important link within the overall communication process.

C. SETUP AND TOOLING SHEETS

Apart from program printed copy and part drawing, *setup sheet* and *tooling sheet* are the other two most important pieces of good program documentation. The major difference between a setup sheet and a tooling sheet is the subject emphasized. Setup sheet is a sketch or a drawing that shows the part layout and orientation on the machine table or in a fixture, possibly even the description of individual operations. Tooling sheet usually lists only cutting tools and their mounting positions, along with spindle speeds, feed rates and offsets for each tool. Examples of both types are shown in this chapter.

One ongoing question many programmers always have is the reminder of a *Catch 22* variation:

'Do I make the setup sheet and tooling sheet before or after writing the CNC program?'

As is usual in many programming applications, adherents and opponents can be found on both sides of the issue. The favorite reasoning for making documentation *before* writing a single line of a program is quite simple. Setup sheet and tooling sheet are the guiding forces for writing the program, the forces of being well organized. Adhering to this method implies a well-organized programmer or a team, it implies that everything is under control. It also suggests that all fixtures and tools and holders and inserts and other tools are already available in the machine shop, ready and waiting to be used. No doubt, and if at all possible, always make both setup sheet and tooling sheet, *before* starting actual program development. Logic behind the reasons for this method is very strong indeed.

Logic itself, however, does not take into consideration machine shop realities, even in cases when they are essentially not the best. A small conflict between departments, a delay in material delivery, tool on back order and similar problems, all contribute to the frustrations of CNC programmers in collecting initial data. Being under pressure from all sides, the programmer has no choice but to improvise, even in times of crisis. Programmer has to *compromise* - in terms of setup/tooling sheet development, it also means a certain downgrading. If there is no choice, always try to find a *reasonable* compromise, but never as an excuse for being sloppy.

The freedom in program development is considerable but it is not unlimited. Normal part program cannot be written without knowing the machine setup and tooling to be used. In many cases, the nature of the job offers many solutions. Even if the *exact setup*, or *exact tool* to be used are not known, think of other ideas, have some opinions - but have ideas and opinions based on some experience (yours or of others). Compromise does not rest with the 'now or later' situation, it rests in the selection of the most *likely* possibility. If something has to be changed, make sure the changes will be minimal. In any case, it is quite possible that the setup sheet and/or the tooling sheet will have to be modified *after* the program has been proven and optimized.

Setup Sheet

In many shops, setup sheets are luxury. It is a simple statement of fact, but many setup sheets are quite poorly prepared if they are prepared at all. Often, they do not reflect the latest program changes and adjustments, they are not consistent between individual machines and even programmers. Although the time spent on preparing a setup sheet is considered nonproductive from the cost angle, it is a time far from being wasted. Setup process can be organized, certain rules can be set and adhered to, and they can be applied to the preparation of a good setup sheet.

The golden rule of a good setup sheet is *to always make it in scale*. Setup sheet using an outline of the material, fixtures layout, finished shape, tool path, etc., should always be done in scale. Scale, even an *approximate* scale, is very important for visual comparison. Clamps and other mounting devices should be drawn in positions corresponding to the actual setup. Tool change location should be marked accurately, different views shown, if necessary. Critical positions should be dimensioned, indicating the maximum or minimum distances.

If cutter radius offset is used, speeds and feeds reflect a certain *nominal* cutter radius. At the discretion of machine operator, cutter radius may be changed within a reasonable range. This range should appear in the setup sheet, including a note on the adjustment of speeds and feeds.

In cases when the cutter exceeds a certain length, it may interfere with the part or other tools. In these cases, setup sheet should include the maximum cutter length allowed within that setup. For chuck work on a lathe, the maximum material jaw grip should be specified in the setup sheet.

Setup sheet has one purpose - to document details of how a particular part is mounted on the machine. It has to cover part holding method and reference point relationships (part, machine, and cutting tool). It has to describe positions of auxiliary devices used, such as tailstock, bar feeder, vise, face plate, hard and soft jaws, and others. A setup sheet template may be useful for each machine. A simple setup sheet is shown in *Figure 49-1*. Feel free to improve it as necessary.

Setup Sheet			
Program reference: ANGLE BRACKET	Date: 07-Dec-08	Programmer: Peter Smid	Sheet: 1 of 1

TOP

FRONT

Figure 49-1

Simple setup sheet form - only basic data shown

A well designed setup sheet should also include information about the material (stock) used for machining, stock the program is based on. Not only the type of stock, also its rough dimensions, amount left for machining, its condition, and other features that are important to include in program documentation. This information is very valuable at its conception and will be even more valuable in the future, mainly for repeated jobs. Many times, a program is made when the stock is not yet available. If the programmer finds out later that there is too much deviation from the estimated conditions, the necessary changes are easier to make with good program documentation.

Although not a strict requirement, some programmers include cutting time for each machining operation on the setup sheet. When a job is run for the first time, the actual cutting time is unknown. As the program is used and optimized, it becomes proven and eventually finalized, the cutting time becomes known with more precision. Knowing the overall cutting time may help in planning the load work on the CNC machine. The most useful cutting time for an individual part is the *chip-to-chip* time that includes all supplementary times (for example tool change time, part replacement time, etc.), not only the cutting time itself.

Tooling Sheet

Although tooling is really part of the setup, it requires a separate set of data, that may or may not fit on the setup sheet itself. If setups and tools used are constantly simple, it may be more convenient to have only one sheet, describing them both. However, for large or complex setups, making a separate tooling sheet is more practical. Both, the setup sheet and tooling sheet, are part of the same documentation and complement - *not replace* - each other.

Machine unit and its CNC system influence the contents of a tooling sheet. Tooling sheet for a lathe will be different than tooling sheet for a machining center. Data gathered for either machine will have some similarities and some unique items. A sample contents of a typical tooling sheet will include description of the following items:

PROGRAM DOCUMENTS

Machine and program identification

Cutting tool type

Tool coordinate data

Tool diameter

Insert radius and the tip number

Offsets associated with the tool

Tool length

Tool projections from the holder

Block number of the tool being indexed

Brief description of the tool operation

Basic speed and feed of the tool

Tool holder description

Tool number and/or tool station number

Special instructions

In addition to the most common items, also include any unique information in the tooling sheet, for example, to inform the operator about non-standard tools, tools that require modification, pre-machined condition of the material, etc. An example of a simple tooling sheet is in *Figure 49-2*.

Tooling Sheet						
Program reference: ANGLE BRACKET		Date: 07-Dec-08	Programmer: Peter Smid		Sheet: 1 of 1	
T#	Tool Description	Tool Ø	RPM	Feed	H..	D..
T01	12 mm Spot Drill	8.7	1650	100.0	H01	---
T02	7.5 mm Drill	7.0	1800	220.0	H02	---
T03	10 mm End Mill - 4 flute	10.0	1210	300.0	H03	D03

Figure 49-2

Simple tooling sheet form - only basic data shown

Coordinate Sheet

The idea of a coordinate sheet is not new. It has been used in programming from its very beginning and it has been mentioned in this digital book many times already. A simple printed form containing the X, Y and Z axes can be used for both machining centers and lathes. *Figure 49-3* shows a good example of a simple coordinate sheet.

The Z-axis column will be usually blank for machining centers and Y-axis column will be blank for lathe programs. Modify the sheet to add additional axes or make separate sheets for each machine type.

Coordinate Sheet			
Program reference: ANGLE BRACKET		Date: 07-Dec-08	Programmer: Peter Smid
		Sheet: 1 of 1	
P#	X coordinate	Y coordinate	Z coordinate
P1	X0.000	Y0.000	

Figure 49-3

Simple coordinate sheet form - only basic data shown

D. DOCUMENTATION FILE FOLDER

All records that have been collected during program preparation are quite likely to be important enough to be kept for future reference. They may be stored all over the place, sometimes very hard to find. So, now is the time to put them all together and organize them. It is time to make a file folder, identify it, fill it up and store it properly.

Identification Methods

Before some better methods of identifying program documentation can be suggested, think about a very popular, yet quite an impractical method. Some programmers use the program number as a reference for all related material. The basic thinking behind this idea is that the available program number range between 1 and 9999 (or even 1 to 99999) will take forever to use up, therefore becomes very useful for some identification purposes. This is a shortsighted thinking, usually by not a very busy programmer who has only one or two machines to take care of.

Look at possible problems with this way of thinking. True, to make almost ten thousand programs (or more) for one machine, it would take 'almost forever'. Even if more machines are available, at a rate of 25 programs a week, numbers will run out in a little more than 7 years. Is that the time to scrap the machine and buy a new one? And if 25 programs a week seems a bit steep, remember that *each* program will have to have a number. That may be three or more separate operations *for a single job*, there may be dozens and dozens of subprograms that also need their own program number. So, the figures are not so unreasonable after all, and some better method should be sought from the beginning. It could be a manually generated method, or a comprehensive computerized database.

A few basic questions remain - what is the best method to employ available program numbers? Is it necessary to employ them at all? Is there an alternative?

The point of this evaluation is that all program number assignment (with the exception of subprograms), should be left exclusively to the *CNC operator*, if at all possible. That means,

another way has to be found to identify the documentation containing all records.

One of the first decisions is the program name selection. Regardless of the number of machining operations or included subprograms, there should be only one folder for one job and only one name for one folder. The name of one folder should share the common denominators with any other folder. Try to make such a name *meaningful*.

With an access to a personal computer, the chances are that all files relating to each program are stored in the computer. In that case, the only limiting factor is software format structure to name the files. For example, the old DOS software files accepted up to eight alpha numeric characters for the file name and another three alpha numeric characters for the file extension. Since Windows 95, long file names are allowed, up to 255 characters plus extension - try to take advantage of this feature. Regardless of which CNC system is used, establish a file naming convention conforming to any possible restrictions. There are several methods for this approach.

One is a totally independent, sequential order. In this simplest form, all documentation related to the first program would be - for example - P0000001, the next program would be P0000002, etc. If the zeros are removed from this format, the files will *not* be listed in a correct alphabetical order on a computer display. No practical limit is imposed here. Another approach would be to use the current drawing number as the basis of documentation identification. This may a good method for the many companies that manufacture their own products, but not much suitable to various jobbing shops. Dealing with many different customers also means dealing with many different types of drawing numbers. The variety may be so great that it is almost impossible to find some common ground for standardization. Another variation on the same theme is a *job number*, rather than a drawing number. In many jobbing shops, each job gets a number assigned the moment the order is received. This *Job Number* is always unique, therefore a good candidate to be used as the number identifying program documentation.

Hopefully, the presented ideas will stimulate many additional ideas that will suit a particular work environment. There are no given rules on the methods of identifying an individual program; there are no rules governing the standard of part program documentation. The old reliable rule is always use the old fashioned *common sense* that is often not so common. Common sense and foresight help any standardization effort. The quality of any standard is measured by its usefulness in the future. For the longer period of time a particular standard can be useful, the better quality of thought has gone into its development.

Operator's Suggestions

When CNC machine operator runs the job, he or she may have comments, ideas, corrections and variety of other suggestions that come from observation of how the program works. It may be a good idea to consider establishing a logbook, a card system, a computer database, or a similar method of communicating the operator's ideas back to the CNC programmer. Whatever system may be selected, it should be available at the machine, so the CNC operator has a primary access to it. The main benefit of such a system is that all communication goes into one source and is easier to keep under control.

Apart from the nature of a particular comment or idea, the log should have the operator's name, current date, perhaps even current time, machine and job description, as well as any other details that may be relevant and useful anytime in the future, particularly when the part is run again.

Filing and Storage

Physical program file folders can be quite bulky, particularly when they contain computer media, such as disks, large size paper drawings, long program printouts, etc. The storage of file folders is usually confined to standard office steel filing cabinets, which should be accessible to *every work shift*, although only qualified and authorized persons should be given the actual access.

If using any kind of computer media for storing the part programs, make sure they are safely stored in a separate container, rather than the file folder itself. Magnetic devices are particularly

sensitive to adverse conditions and should be stored away from any heat source and magnetic field (including a telephone, for example). They should be kept in a dry and dust free environment. Keeping duplicates (or even triplicates) in a separate place is also a good and safe procedure.

Much more practical - and much less bulky - than physical paper is storage of proven programs is storage on a CD (Compact Disc) or a DVD (Digital Versatile Disc), using a standard hardware and software. Although these disks still require to be stored away from all heat sources, they are not a subject of magnetic fields and take much less space. Incorporating Adobe PDF (*portable document format*) files is another reliable option that saves space.

In physical storage, individual sheets or pages of the part program documentation should be either numbered consecutively or have a reference number on each page. Drawers of the filing cabinets should be identified as to their contents. These are common enough requirements, but very often ignored all together, usually *because there is no time*. The main philosophy behind an orderly filing system is the speedy access to a required program that provides instant and accurate information.

36. PROGRAM VERIFICATION

When a CNC program is completed, there will be a written copy, or a file copy stored somewhere on a computer. At this point, the program development is fully completed. It may be a perfect program with no errors. Of course, that was the intent from the beginning - to make an error free program. What happens if - *in spite of the best efforts* there is an error? Even a small typing error can cause a severe problem when the program runs on the machine. Could an error be prevented? And if so, how?

Every program should be checked against all predictable errors *before* it reaches the machine. Checking can be quite simple, such as a visual comparison of the written copy and the printed copy. The main purpose of a program check is to detect obvious mistakes - mistakes that can be seen by *concentrating* and *looking for them*. The kind of errors detected first are mostly syntax errors. Of course, there is no guarantee that the program is error free, but when it does leave the programmer's desk, the effort should be to *make it* error free. All programs arriving at the machine should gain the confidence of CNC operator. Operator should be free to concentrate all efforts on proving the program sequences and run the first part. CNC operator has no time to check for program errors that could - and should - have been detected in the 'office'. To do all program proving on the machine is nonproductive and should be avoided.

A. DETECTION OF ERRORS

Before an error of any kind can be corrected, it must be discovered first. In any type of programming, errors can be found either *before* or *after* the program leaves programmer's desk. Intentions of any professional programmer are undoubtedly to discover any errors *before* they are detected during program execution, at the CNC machine. This is a *preventive* effort. If an error has to be corrected at the machine, during actual program run, CNC operator has to do something that should not normally be part of his or her job. Whatever action it is the operator must take; it is always a *corrective* action. Therefore, there are two measures that can be taken to help eliminate errors in CNC programs:

Preventive measures ... *proactive measures*

Corrective measures ... *reactive measures*

Preventive measures are those that all parties involved should participate in with suggestions and constructive criticism. On the other hand, corrective measures require certain skills, knowledge, and even some authority.

Preventive Measures

All errors should be detected and corrected by the programmer, who has taken a certain number of preventive measures. The first preventive measure is to *get organized*. Set up *procedures*, set up *standards*, set up *rules*. Then, follow them diligently. Errors that can be found before a program is used on the machine are numerous. Yet, it takes unique techniques to become successful in their detection.

The first method any programmer should use is simple *check your own work*. Read the program and *evaluate* it. If the rules of consistency had been followed, the error check is easy. Programmers know the appearance of a program, its structure, the established standards, the order of various commands at the beginning and end of each tool. Checking the program itself should not take much time at all.

The second method can be successfully applied when working with other programmers or skilled operators. Ask a co-worker to look through any new or changed program. Do not be surprised to see mistakes such a check can reveal. A fresh, detached, and impartial look can be very productive. Sometimes, even taking a little break or fresh air first will rejuvenate the brain cells.

With a computer and a special simulation software, such as the enclosed *NCPlot* (on CD), there is a third option make a graphic representation of the toolpath on the screen.

A major part of preventive measures is finding *syntax errors*. A syntax error is one that can be detected by the control unit. For example, if a dollar sign appears in the program, the control will reject it as illegal. The control returns an error message or an '*alarm*'. If the digit 2 is entered in the program instead of the intended digit 7, that is *not* a syntax error. That is a *logical error*, since both are legitimate characters the control can accept.

Corrective Measures

If an error is discovered at the control, the preventive check did not uncover it. An error that is found at the machine slows down production. It forces the machine operator to take corrective measures and eliminate such error. Operator can take one of two actions. One will be to return the program to the programmer, another action will be to correct the error at the machine. Which choice is better depending on the error seriousness. An error can be *soft* or *hard*. A soft error is one that does not require stopping the program from being processed by the CNC system.

For example, a missing *coolant* function M08 in the program can be switched on manually at the machine, without interrupting program processing. That is an example of a soft error - it is still an error but classified as a *minor* error.

A hard error occurs when the program processing must be stopped by the operator, *as the only available choice*, and without doing a damage to the machine, cutting tool, part, or all of them. A common example of a hard error is a programmed tool motion that cuts in the wrong direction. Program itself is wrong and must be corrected. This is an example of a hard error, classified as a *major* error.

Most CNC operators do not like delays, especially delays caused by somebody else. A dedicated machine operator will do anything possible to correct a problem without any assistance. For program errors, the operator will try to take corrective measures to eliminate the problem. Not every operator is *qualified* to do even a simple change to the program. On the other hand, some qualified operators may not be *authorized* to do program changes as a matter of policy.

Every company benefits greatly, if the CNC operator has *at least* a basic training in CNC programming. The purpose of such a training is not to make the machine operator a fully qualified CNC programmer. Its purpose is to highlight how a part program influences CNC machining, the setup, tooling and all the other relationships between programming and machining. Its purpose is to offer the operator tools that can be used for minor program changes, etc. Such a training, if it is designed and delivered in a professional manner, is always a worthwhile investment. It may be a relatively short training that will pay for itself very quickly. Time delays on CNC machines are costly and the sooner the program is made functional, the less damage to the production control has been done.

Whenever a program has been changed at the machine,

the program documentation must reflect these changes, particularly if they are permanent. Even a small permanent change should always be documented in *all* copies of program documentation.

B. GRAPHIC VERIFICATION

Programming errors can be costly, even if their cause is a minor human error. Omitted minus sign, a misplaced decimal point, an illegal character - all are minor oversights that cause major errors. Although a visually checked program should be error free, that may not always happen. The human eye is weaker when it evaluates non-graphic elements.

One of the most reliable methods of part program verification is a graphic display of the toolpath as it appears in the program. Almost all errors relating to the toolpath can be detected early, by one of three available graphic verification methods.

One method of graphic verification of a CNC program is a screen plot. This *optional* control feature will show all programmed tool motions on the control screen. Tool motions will be represented as lines and arcs. Feed rate motions will appear as a solid line of the selected color, rapid motions will appear as a dashed line. Display of the toolpath will appear on the control screen.

Many controls offer a graphic simulation option, where the toolpath is simulated on the screen. Each cutting tool can be shown by a different color or density, making the visualization easier. Some graphic simulation uses actual tool shape and the part for a realistic display. The negative part of any graphic verification is that it can only be used when the program is loaded into the control.

The second verification method is much older than the first. It is a hard copy plotted representation of all cutting tool motions. Hard copy plotting has been available in computer programming for a long time. To get the benefits of hard copy plotting, a pen plotter and a suitable software will make it work. Plotter is seldom a problem in companies using CAD software but may not be available to small machine shops. Required software is also part of a large computer based programming system and can be quite expensive. A simple version of a pen plotted toolpath is a screen dump, usually to a printer. Pen plotting in CNC is not a common activity.

There is a third method of graphic verification and can be off machine. It uses a computer software designed to read a manually generated program, then displaying it on the computer screen. Typical software of this kind is *NCPlot* - software with many advanced features.

C. AVOIDING ERRORS

The goal of every programmer is to write error free programs. That is almost impossible, since any human activity is subject to occasional errors. Programmers with all levels of experience make mistakes, at least once a while.

Since prevention of errors should be the main goal of any programmer, this section looks at the subject in more depth. Some most common mistakes will be evaluated, along with suggestions to prevent, or at least to minimize, their happening. First, what exactly is a program error?

Program error is the occurrence of data in a program that will cause the CNC machine to work contrary to the intended plan or not to work at all

All errors can be classified into two groups:

Syntax errors

Logical errors

Although the average distribution of programming errors could be generally split at 50/50 between syntax and logical errors, certain conditions may swing the balance. A programmer with limited experience will make all kinds of errors. An experienced programmer makes more syntax errors. Let's look at each error group.

Syntax Errors

Errors in this group are usually easy to deal with, once they are identified. Syntax error is simply one or more characters in the program that are either misplaced or do not belong there. This error covers program entries that do not conform to programming format (known as *syntax*) of the control system. For example, two-axis lathe control systems do not accept the character Y. If such control encounters the letter Y in a lathe program, it will reject it as a syntax error and the program won't run. The same result will happen when the letter U is programmed for most milling controls. Some other letters cannot be used with either system - for instance the letter V - it is an illegal character for most milling and turning controls. Yet, it is very legal character in a four axis wire EDM control.

Syntax errors also occur if valid characters are used in combination with an option not supported by the control. A good example is a custom macro, an option on most Fanuc controls. Custom macros use a number of standard letters, digits and symbols, but also a number of special symbols, for example a sharp sign #, brackets [], asterisk *, etc.

Macros also use some special words, such as COS, SIN, GOTO and WHILE, words not allowed in a standard program. Macro symbols or words in a non-macro program result in a syntax error. The error will also occur when the custom macro option is available, but some of the symbols are used incorrectly or the special words are misspelled.

Control system handles syntax errors arbitrarily - *it simply rejects them*. This rejection is displayed as an error message on the screen. Program processing will stop. Syntax errors are irritating and embarrassing, but almost harmless. Scrap as a result of a syntax error is possible but rare. The second group - the *logical* errors - is much different.

Logical Errors

Logical errors are more serious than syntax errors. A logical error is defined as an *error causing the machine tool to act in a way contrary to the programmer's intentions*. If a motion is programmed to the absolute coordinate of X1.0, but program states X10.0, the control will go ahead but the tool position will be wrong. The same error will happen when Z10.0 is programmed, although the intent was X10.0. The control *does not* and *cannot* have any built-in protection against logical errors. Programmer has the responsibility to exercise all necessary care and caution. Logical errors can be serious - they may not only result in a scrap, they can damage the machine and even harm the operator.

Logical errors cover an unlimited number of possibilities. For example, the following lathe program is *wrong*:

O5001

(EXAMPLE WITH ERRORS)

N1 G20 G40 G99

N2 G50 S2500 T0400 M42

N3 G96 S530 M03

N4 G00 G41 X12.0 Z0.1 M08

/ N5 G01 X-0.06 F0.012

/ N6 G00 Z0.2

/ N7 X12.0

N8 Z0

N9 G01 X-0.06

N10 G00 Z0.1 M09

N11 X20.0 Z5.0 T0400 M01

...

There are three errors in the O5001 example. Try to identify them before reading any further.

The first error should be easy - a *tool offset* is missing. In block N2, tool T0400 is selected without an offset. This block is correct. Block N11 is the return to the indexing position, with expected tool offset cancellation, which was never programmed. The error is in block N4 - it should be:

N4 G00 G41 X12.0 Z0.1 T0404 M08

The second error is rather hidden and requires a trained eye to spot it. Note that a block skip symbol was used in blocks N5, N6 and N7. When running the program with the block skip setting ON, cutting feed rate will be missing in block N8. In this case, the control would issue an error message, but only for the *first time* this program is processed. The correct block N8 should be:

N8 Z0 F0.012

The third error is the missing cutter radius offset cancel in block N11. This block should correctly be written as:

N11 G40 X20.0 Z5.0 T0400 M01

An error of this kind may have a serious implication for the next tool. Even worse, this error may not be discovered during the first part run. The correct program is O4902:

O5002

(EXAMPLE WITHOUT ERRORS)

N1 G20 G40 G99

N2 G50 S2500 T0400 M42 N3 G96 S530 M03

N4 G00 G41 X12.0 Z0.1 T0404 M08

/ N5 G01 X-0.06 F0.012

/ N6 G00 Z0.2

/ N7 X12.0

N8 Z0 F0.012

N9 G01 X-0.06

N10 G00 Z0.1 M09

N11 G40 X20.0 Z5.0 T0400 M01 ...

After evaluating these logical errors, what chances are there that the control will detect them and return an error message? *Nil, zero, zilch*. All errors in the example are good illustrations of logical errors. They may not always be easy to find, but they can create additional problems if not found early.

D. COMMON PROGRAMMING ERRORS

Strictly speaking, there are no '*common*' programming errors. Every programmer makes some unique mistakes. It is difficult to list any errors as being more common than others. It is also true, that some mistakes are made more frequently than others and in that sense they are more common. Focusing on this group should be beneficial.

Both syntax and logical errors share the same cause - *the person who writes the program*. The most important step towards eliminating errors is the identification of a problem - ask yourself '*what mistake do I do repeatedly?*' Everybody makes some 'favorite' mistakes, the solution lies in the correct answer to this simple question.

Most errors are results of insufficient program planning and a lack of precise programming style. Planning offers a sense of direction; style offers tools and organization.

The simplest - *and the most frequent* - error is an omission of some fundamental instruction. It

may be a coolant function, program stop, a missing minus sign and others. Even the whole block may get lost, mainly when preparing the program from poor sources. Many errors are caused by the programmer's inability to *visualize* what will exactly happen when the program is processed. To this category belong all errors relating to setup, tooling and machining conditions - cuts that are too heavy or too light, insufficient clearances and depths, incorrect spindle speeds and cutting feed rates, even the selection of wrong tools for a given job.

Program Input Errors

Most programs are handwritten or typed and have to be transferred to the control system or a computer file. Many errors are caused by the *incorrect input of intended data*. Keep in mind that if somebody else is using the program, its legibility (if hand-written) and syntax is very important.

Input errors also include errors caused by forgetting to input significant characters in the program. These strings can be almost anything and can cause a serious problem. A missed coolant function is not likely to cause a big problem; a missed decimal point or a wrong tool retraction will. Other errors are insufficient tool clearances, a depth that is too shallow or too deep, errors relating to cutter radius offset (this is always a big group). Be also careful when canceling or changing modal program values. One common error is to cancel one kind of motion by replacing it with another type of motion in one block, then forgetting to reinstate the previous motion later.

Calculation Errors

Using math functions and formulas is an integral part of developing CNC programs manually. Calculation errors include a wrong numeric input, even when a pocket calculator is used. Keying a wrong formula, wrong arithmetic sign or placing parentheses in a wrong position, all represent a serious error.

Rounding Error

A special type of an error is caused by *incorrect rounding*. This error is an accumulative error that results from too many dependent calculations. A rounded value used in other calculations may lead to an error. In many cases the error will be too small to cause any problems, but never count on it. It may become a very bad habit.

Calculations check

To prevent math errors when using formulas for calculations, it is a good idea to check the calculated result once more, using a *different* formula. Math is a generous subject, and more than one calculation method is usually possible.

Hardware Errors

The last type of program errors is by the *malfunction of a hardware element* of control system or machine. In CNC, even a bug in the software is possible. Their occurrence is rare, as modern controls are very reliable. When encountering an error, don't blame the control or the machine as *the first and only possible cause*. It shows a degree of ignorance, arrogance, and unwillingness to address the problem responsibly. Before calling for a service, make sure to exhaust all other possibilities of error detection first.

Miscellaneous Errors

Some errors can be traced to the part drawing. An error in a drawing is possible, but first make sure to interpret any drawing correctly. Drawing errors include too many or too few dimensions, poor tolerances, non-matching dimensions, etc. Also make sure to use the latest drawing version.

Other errors may be caused by the wrong setup, tooling or material. These are not programming errors, but they have to be considered as possibilities. With some common sense and suitable precautions, many programming problems can be eliminated. For example, to prevent an unproven program to be processed as a proven program, just mark it as unproven. Mark it at the *beginning* of the program and leave it there until the program is checked.

A complete elimination of errors is not realistic. Mistakes do not happen - *but mistakes are always caused*. Inexperience, negligence, lack of concentration, poor attitudes, are just some causes. Always program with the attitude to eliminate programming errors altogether. That will be the first step to making fewer errors.

37. CNC MACHINING

When a part program is fully completed and sent to the machine shop, programming process is over. All calculations have been done, program has been written and well documented, the program file is on the way to the CNC machine. Is the programmer's job really finished? Is there some reason that could bring the program back, perhaps with operator's comments, suggestions, or even criticism?

If the delivered part program is *perfect*, programmer will not hear a word from any direction. No doubt, programmer will hear negative comments from *all* directions. The question is - when is the programmer's responsibility *really* over? At what point in the process of manufacturing can the programming results be evaluated? When can the program qualify as a *good* program?

Probably the fairest and the most reasonable answer would be *whenever the part has been machined under the most optimized working conditions*. This means that the programming responsibility *does not* end with the program and documentation delivery to the shop. The program at this stage is still very much in the development process. It still has to be loaded to the CNC system, the machine has to be set up, cutting tools mounted and measured and a variety of small jobs done before the first part can be started. True, all these tasks are the responsibility of CNC operator, so there is no need for the programmer to care what happens during machining, right?

Wrong! Every CNC programmer should make an effort to be in constant touch with the actual production. In the field of business software development, it is quite normal to have a team of people to work on a certain large programming project. After all, most programming ideas come from talking to colleagues and the actual users of the program or particular software. The same is true for CNC programs used in machine shops. The users of programs are typically CNC machine operators - they can be a gold mine of constructive ideas, improvements and suggestions. Talk to them, ask questions, make suggestions, and - most important - *listen* to what they have to say. Programmers who never put their foot in the machine shop or go there reluctantly, programmers who may go there with their eyes closed and ears plugged, programmers who take the attitude that they are always right, are all on the wrong track. Exchanging ideas with machine operators, asking questions and seeking answers is the only way to be fully informed about what is actually going on in the machine shop. It is in programmer's interest to know how the CNC operators *feel* about the program, the programming style and the approach to programming overall. *Do* exchange ideas and *do* communicate with each other - that is the best advice for becoming a better CNC programmer. Machine shop offers tremendous resources, take advantage of them.

CNC technology is an instrument to improve productivity with a minimal human involvement, measured at least by the physical level. As any other technology, it must be managed intelligently and by qualified people with experience. Without a firm grip and good control, without good management, the technology will not yield the expected results - in fact - in will become counterproductive.

The function and responsibilities of a CNC programmer has been covered. Now, let's look at what happens when the completed program and related material actually reach the machine shop.

A. MACHINING A NEW PART

The most expensive part done on a CNC machine is always the *first one* of the batch. After the machine setup is completed, CNC operator is ready to test the program and machining conditions. Setup time is always non productive and testing a program is non productive as well. It takes quite a bit of time and effort, even if a good part comes out of the first run, as it should. These activities are necessary and must be done but doing too many 'first' parts for one batch is not productive either.

Generally, there are two groups of CNC programs, each having a different effect on program proving. The first group covers all programs that have *never* been used on the CNC machine. These

programs must be proved for accuracy, as well as optimized for best performance. The second group covers the *repetitive* jobs - programs that have been used at least once before and have been proved to be correct in all respects. Programs in this group have most likely been optimized for the best performance under certain given conditions. In both cases, the CNC operator must take a good care when running the first part of a batch. However, there are differences between a *new job run* versus a *repetitive job run*.

In either case, two qualities relating to the part program have to be established first:

Setup integrity

Program integrity

These two considerations are equally important - if only one of them is weak, the final result is not satisfactory. Always aim at the highest level in either category. Also keep in mind that the setup integrity has to be established again with each run in the future. Program integrity has to be established correctly only once.

Setup Integrity

Machine setup is only a general description of the type of work actually done to get the CNC production going. The whole process covers setup of cutting tools, as well as part setup and many related tasks. No single check list can ever cover all points that have to be considered during a CNC machine setup. The major focus here is at the most important considerations, in form of a brief check list. Adjust the individual points according to the machines and CNC systems in your shop. Adjust the list to reflect personal working methods and/or programming style. The main purpose of this check list, or any other for that matter, is to cover as many details as possible and not to omit an important item, operation, procedure, and so on. Even a small omission may cause an accident and part damage or even a scrap due to a faulty machine tool setup.

Cutting Tools Check

Are the tools properly mounted in holders
Are the proper inserts used (Radius, grade, chip breaker, coating)
Are all the tools the right size
Are the tools placed in the proper magazine station
Are the offsets set correctly (set zero to unused offsets)
Is there an interference between individual tools
Is the boring bar properly oriented (milling)
Are all the tools sharp

Part Setup Check

Is the part mounted safely
Is the part properly oriented on the table (milling)
Is the projection of the part from the chuck safe (turning)
Is the part lined up for squareness (milling)

Are the clearances sufficient
Are all the clamps away from the cutting path
Is the machine at its start (home) position before you press Cycle Start
Does the tool change take place in a clear area

Control Settings Check

Is the coordinate setting registered (for G54 to G59)
Are all the offsets entered correctly
Is coolant necessary
What is the status of the BLOCK SKIP switch
Is the optional program stop M01 active (ON)
Is the DRY RUN off if the part is mounted
Do you start with a SINGLE BLOCK mode set to ON
Do you start with spindle speed and feed rate overrides set to LOW
What is the status of MANUAL ABSOLUTE switch (If applicable)
Has the position read-out on the screen been set from zero (origin preset)

Machine Tool Check

Is the slide lubrication container filled with proper type of oil (lubricant)
Is the coolant tank filled
Is the chuck and tailstock pressure set correctly(turning)
Has the machine been zeroed before running a job - is the read-out set to zeros
Is there enough pressure for the air attachments (air hose, etc.)

Program Integrity

Any new and unproved program is a potential source of problems. In manual CNC programming, mistakes are a lot more common than in CAD/CAM programs. A good way to look at a new program is through the machine operator's eyes. Experienced CNC operators take a direct approach when running a new program - *they take no chances*. That does not mean the CNC programmer is not to be trusted - it simply reflects the fact that the machine operator is ultimately responsible for the expected quality of work and is aware of it. He or she has a sense of great responsibility. Whether the damage to the part or even scrap is caused by the program or for some other reason is a little consolation when the work is rejected.

What does the CNC operator look for in a new part program? Most machine operators would agree that the first and the most important thing is *consistency* in programming approach. For example, are all tool approach clearances the same way as always? If not - is there a reason? Is the basic programming format maintained from one program to another program and from one

machine to another? A good operator scans the written program *twice* once on the paper copy, the second time when the program is loaded into the control system. It is surprising what can be seen on the screen that was not seen during the paper copy check. Reverse is also true. Common mistakes such as a missing minus sign or an address, a misplaced decimal point or a programmed amount extra-large or extra small can usually be detected on the screen easier than on paper. If using a computer for manual programming, print out the program and check it visually. Using a double check, many costly mistakes can be prevented. There is software available to graphically check the program on a computer, using toolpath simulation. *NCPlot* is such software and is included as a shareware on the CD in this digital book.

Consistency in programming style is very important and cannot be overstated. Consistency is an important way to gain confidence of CNC operators in program integrity.

B. RUNNING THE FIRST PART

CNC machine operator usually starts a new job by studying the documentation included with the program, mainly drawing, setup sheet and tooling sheet. The next few steps describe a standard setup procedure that will vary occasionally, but they will remain the same for most jobs.

STEP 1 - Set the cutting tools

This first step uses a tooling sheet or tooling information from the part program. CNC operator sets the cutting tools into their holders and respective tool stations and registers all tool numbers into the control memory. Make sure the tools are sharp and mounted properly in the holders.

STEP 2 - Set the fixture

A fixture that holds or supports the part is mounted on the machine, squared and adjusted, if necessary, but the part itself is not mounted at this point. Setup sheet serves as the documentation, particularly for complex setups. A fixture drawing may often be required as well.

STEP 3 - Set the part

Locate the part into the fixture and make sure it is safely mounted. Check for possible interferences and obstacles in the setup. This step represents the end of most initial steps of CNC machine operation.

STEP 4 - Set the tool offsets

Depending on the type of machine, this step takes care of setting the tool geometry and wear offsets, tool length offset and cutter radius offset, if applicable. One of the most important parts of this step is the setting of work coordinate system (work offsets G54 to G59) or the tool position registers (G92 or G50) for old controls, but *not* both, even if they are available. Work offset setting is by far the best and most convenient selection of modern CNC machine tool setup.

STEP 5 - Check the program

This step is the first evaluation of the part program. The part may be removed from the fixture temporarily. Since all offsets are already set in the control, program is checked accurately, with all usual considerations. Program override switches on the control panel may be used, if required. Watch for tool motions in general and be sure to watch for tool indexes specifically. Repeat this step, if not absolutely sure with any aspect of the programmed toolpath.

STEP 6 - Reset the part

If the part was removed in the previous step, now is the time to mount it in the fixture again. Successful completion of all previous steps allows continuation with proving the first part. At this point, check the tooling once more, also check the oil and air pressure, clamps, offsets, switch settings, chucks, and any other important machine feature, just to be sure.

STEP 7 - Make a trial cut

An actual trial cut may be required in order to establish whether the programmed speeds and feeds are reasonable or not and if the various offsets are set properly. Trial cut is a temporary or an occasional cut that is designed to identify minor deviations in actual offset settings and allows their change. Make sure the trial cut leaves enough material for actual machining. Trial cut also helps to establish tool offsets to keep dimensional tolerances within limits.

STEP 8 - Adjust the setup

At this point, any necessary adjustments are finalized in order to fine tune the program before production begins. This step includes final offset adjustment (usually a wear offset). It is also a good time to adjust spindle speeds and feed rates, if necessary.

STEP 9 - Start the production batch

A full batch production can start now. Again - a quick second double check may prove to be worth the time.

The ideal way to run a new program is to run it first through the control graphic display, if available. It is fast and accurate and offers a lot of confidence before actual machining. This test can be done with a variety of override modes in effect, for example, *Machine Lock* or *Single Block*. Do not underestimate features such as *Zero Axis Neglect* and *Dry Run*, when testing a new program.

If using graphic options of the control system, most likely, there will be two kinds of graphic representation of the tool path:

Tool path simulation

Tool path animation

Both features have been described in the last chapter.

The first type of graphic representation, *tool path simulation*, shows the outline of a finished part and the tool motions. Part outline is identified by a single color, the tool motions are identified by a dashed line (rapid motion) and a solid line (cutting motion). During program processing, the order of machining is shown on the display screen as either dashed lines or solid lines, depending on the motion type. The solid area of the part is not shown, neither are the tools, chuck or tailstock for lathe applications. With a color screen, the colors can be preset for each tool to further enhance the flexibility of graphics display.

The more descriptive method of verifying the toolpath prior to machining is *tool path animation*. In many respects similar to the toolpath simulation, the toolpath animation offers a few additional benefits. Part can be seen as a shaded form, rather than an outline only. The tool shape can be preset and seen on the screen display; the chuck shape and size can also be preset, as well as the outline of tailstock, fixture, etc., all in shaded form. The result is a very accurate representation of the actual setup conditions. As an additional benefit, the display is also proportional in scale. During the actual cut, material can be seen as being 'machined', right on the screen. Toolpath animation is a significant improvement over toolpath simulation.

Do not expect 100% accurate display of any toolpath. No graphic display can show every single detail and no simulation will show flying chips. What it does show is quite impressive, however. For CNC machining centers, the control with graphics can be set to one of several selectable views. More than one view can be set at the same time on the display screen, using a split screen method, also called *windows* or *viewports*. Many CNC operators run the graphic display twice, especially for milling systems - once in the XY view, the second time in the ZX or YZ view and even isometrically (XYZ view). Make sure the rapid motion display is turned on. Display can be run in a single step mode, areas that are either too small or especially critical can be enlarged (or reduced for large parts) with zooming features. Cutter radius offset, tool length offset, and other functions can be turned on or off for the graphic display. Make sure the simulated conditions are as close to real conditions as possible. Also, do not forget to have all tools and offsets set *before* the program

is tested. Unfortunately, these graphic options also add quite a bit to the overall cost of the control system and many companies choose not to purchase them.

Many programming instructions cannot be tested by using the graphics only. On most controls, there will be no clamps, no spindle speed or feed rates. Many other important activities cannot be seen, but what does show will make the actual cutting so much easier. Since all motions have been tested in the graphic mode of the control, all that has to be done during the actual run, is to concentrate on those details that could not be seen on the display. The tasks to be checked have been narrowed down and the part program is easier to follow.

C. PROGRAM CHANGES

Even if a part program is proven, tested and the first part made and inspected, a good CNC operator looks at ways of improvement. Some improvements may be done immediately on the machine, before the whole job is completed. Some improvements may require a different setup, tooling or fixturing. Often, it would not be practical or even possible to implement those changes on the current job, but they should be applied the next time the same job is done. Some changes to the part program are result of a design modification and have nothing to do with program optimization. Others are strictly steps taken for the best productivity results. Regardless of the reason, virtually any change required by the machine shop involves the CNC programmer who has to apply any new changes to the new program.

All changes to a program should be for the better, they should *improve* the program. Often a major change will require a complete program rewrite, but more likely, a program can be modified to a reasonable extent. When a program is changed for the better, it is said to be *optimized*, it is upgraded. This method can be compared to another type of program change - *program update*.

Program Upgrading

Upgrading a CNC program means to *strengthen* it, to *enrich* it, to make it *better* than it was before. It means to change it in a way that the cost of part production is significantly decreased. The cost reduction must be achieved with no compromise in quality of the part or machining safety.

The most common form of program upgrading (optimization) is minor changes to spindle speeds and feed rates. The process is called the *cycle time optimization*. Milling operations may require a different approach than turning operations. Jobs that are repeated frequently, as well as large size lots, should be scrutinized with even more care. Keep in mind that only *one second* saved on a cycle time will save one hour for each batch of 3600 pieces, half an hour for each 1800 pieces, and so on.

In the following check list are some major points to consider when optimizing a CNC program. The list is far from complete, but it should serve as a guide to what areas can be looked into and be explored. Some items in the list apply only to the milling operations, others only to turning. There are also some items that apply to both systems. Several of them require a special option of the control system or the machine tool to be available.

Fine-tune the spindle speed and/or feed rate
Choose the heaviest depth of cut possible
Choose the largest tool radius possible
Experiment with new cutting materials
Rearrange tool order for faster tool changes
Program bi-directional turret rotation

Let one tool do as much work as possible
Use M01 rather than M00 whenever possible
Avoid excessive dwell times
Eliminate 'air cutting' situations
Shorten rapid motions where applicable
Use multi-axis motion whenever safe
Apply fewer passes for threading
Look for block skip applications
Avoid spindle direction change
Shorten the tailstock travel distance
Do not return to machine zero after each piece
Program tool changes close to part
Reassess the setup and/or design a new one
Re-evaluate your knowledge and skills
Consider upgrading the CNC system

This check list is a typical sample only, although developed from experience. Many more items can be added to this list and many item can be modified in their description. Even programs that are to be used only once should be carefully audited. There may be an improvement that can be applied to a different job, sometimes in the future.

Program Updating

In contrast to program upgrading (optimization), the reason for program updating has nothing to do with decreasing the part cost. In the end, the part *may* cost less, due to a change in engineering design or similar interventions, but not because of a program change. A program needs to be updated *after any change is made in the drawing that affects the CNC machining*. Even programs that have been previously upgraded may still have to be updated.

Engineering changes in part design are more common in companies that manufacture their own product line. In a job shop, the design changes are typically initiated by the customer, but have the same overall effect. The only difference is in the source and origin of such change.

A specific change that will affect the upgrade of a CNC program may be as small as a change in a single dimensional tolerance or as large as a complete part redesign. Personal experience may be somewhere between the two. An upgraded CNC program will reflect the magnitude of the change - whether it is a minor correction or a complete program rewrite.

Documentation Change

The documentation that is associated with a particular part program is not much useful if it does not reflect all program changes done during part machining. Just like a well-documented engineering drawing or other important data source, all revisions, updates, upgrades and many other changes should be recorded. Changes in mathematical calculations should be especially well documented and supplemented with formulas and sketches if possible. If there are several existing copies of the documentation, they too, should be replaced to make them current and up to date.

The programmer's name, the nature of the change, the date, even the time of the day, should be used to indicate *when* such a change took place. Keeping the old version for reference (at least for a while) may also be a good idea. Sometimes one or two experiments may be necessary before deciding on the best documentation, on the final documentation suitable to particular needs.

D. ALTERNATE MACHINE SELECTION

Even with the best planning, things can go wrong, at least occasionally. What happens in a machine shop when the only CNC machine is suddenly out of commission? Of course, this never happens, except when a rush job is just about to be set up on that very machine. It usually happens when it is expected the least.

Every production manager has to have an alternate plan of action. One of the most common actions is to do the job on another CNC machine. Of course, such a machine has to be available, but there is more to consider.

Usually, a job is programmed for a specific machine and a CNC system. If two or more such machines have been installed in the shop, the program can be executed on any one of them. Comparably, if two or more machines and/or controls are totally incompatible, programs are not transferable, and a new program must be developed. The best opportunity for compromise exists if two machines are different in size, but with the same control type. The existing program may be usable as is, or with only very minor modifications.

The major considerations for alternate machine selection involve tooling and setup. First, the cutting tools and holders, as well as fixtures, must be available. Tools must all be the same size, even if holders are different. Part position on the table, clamps locations, data holes, clearance areas, etc., must also be the same. In addition to these general considerations, specific conditions such as spindle speed, feed rates, power rating of the machine tool and other factors must also be carefully examined. Overall accuracy and rigidity of the alternate machine is also very important.

Those machine shops where many part programs have to be portable have adapted various standards for both programming and setup operations.

E. MACHINE WARM UP PROGRAM

Any precision equipment is guaranteed by its manufacturer to work accurately not only if it is handled properly, but also if it operates within a certain environment. Computers - CNC systems included - are particularly sensitive to rapid changes in temperature, humidity, dust level, external vibrations, etc. All potential hazards are clearly specified in the manufacturers' literature. Every CNC operator knows from experience that the part precision depends a great deal on the spindle temperature. Some ultra-high precision machines even have an internal cooling system to keep the spindle temperature constant. In cold climates, on a cold morning in the winter, when the machine was sitting all night in an unheated shop, experienced CNC operator turns the spindle on for a few minutes, to let it warm up. At the same time, in order to make the slide lubricant freely moving along the guide ways, the operator makes a few free motions in both directions of all axes. If this process is repeated every day in cold winter months, it may be worth to automate it. A short program will do the job.

To write such a program is simple, but there are several important points to consider. First, make sure that the machine motions will always be in the area where there is no possibility of a collision. This program will be used with many jobs and modifying it every time a new job is set up is not an option. Another point to consider is the spindle speed in r/min. Avoid programming an excessively high r/min - a tool mounted in the spindle for the warmup could have a small or large diameter. To make the program to repeat itself indefinitely, use M99 function at the end. Program also function M30 for the program end, but with a block skip symbol [/]. When the warmup is to terminate, simply turn the block skip switch off. All machine motions will be completed, and the program will end naturally.

Example O5101 is a typical warm up program for a milling system and uses imperial units. The program can be easily adapted to any other machine:

O5101 (WARM-UP FOR A MILL)

N1 G20

N2 G40

N3 G91 G28 Z0

N4 G28 X0 Y0

N5 S300 M03

N6 G00 X-10.0 Y-8.0

N7 Z-5.0

N8 S600

N9 G04 P2000

N10 X10.0 Z5.0

N11 Y8.0

N12 S750

N13 G01 X-5.0 Y-3.0 Z-2.5 F15.0

N14 X-2.0 Y-2.0

N15 Z-2.0 S800

N16 G04 P5000

N17 G28 Z0 M05

N18 G28 X0 Y0

/ N19 M30

N20 G04 P1000

N21 M99 P5 (REPEAT FROM BLOCK 5)

%

This example is simple in structure, yet well thought out. There are several intentional programming techniques contained in the sample program:

Whole program is in incremental mode

First motions are to the machine zero

Z-axis motion is the first motion

Spindle speed is increased gradually

Dwell is used to lengthen the current action

End tool motion is to the machine zero

End of program M30 is 'hidden' by a block skip function

Each program repetition starts at block N5

Several program versions can be developed, depending on the machine and the type of work expected on that machine. For example, if developing a warmup program for a CNC lathe, incorporate functions that are typical to a CNC lathe - for example, changing the gear range,

moving the tailstock in and out, opening and closing the chuck jaws, doing a tool change, etc. On a horizontal machining center, include the indexing table motion; on a boring mill, the in and out spindle quill motion may also be programmed. Modify the program to suit any particular purpose, but keep in mind its aim - to warm up a machine that had been idle for a relatively long period of time in a cold temperature. Also keep in mind the safety of operations - the goal is *a generic program for a specific machine type*, a program that can be used with all jobs, without modifications.

F. CNC MACHINING AND SAFETY

Machine shop safety is everybody's responsibility. Some basic safety issues have already been introduced in the first chapter of this digital book. CNC programmer has to apply safety at the programming level, CNC operator at the machine level, and so on. Many companies have established numerous safety rules and procedures that work well. Follow them and try to improve them.

Generally, the safety concerns of a machine operator are almost the same as those operators running conventional equipment. Safety starts with a clean workplace and organized approach to programming, setup and machining. Many do's and don'ts can be itemized, but no list will satisfy all the safety concerns. Here is an attempt at a typical list of safety concerns in CNC shop. There are several general groups in the incomplete list. Many suggestions can be in different groups.

Personal Safety

Wear suitable clothing (tucked-in shirt, buttoned-up sleeves)

Remove watches, rings, bracelets, and similar jewelry before machine operation

Keep long hair under a net or tied up

Protect your feet by wearing approved safety shoes

Protect your eyes - wear approved safety glasses with protective side shields at all times

Wear an approved safety helmet if that is the company policy

Always protect your hands - never reach towards the part while the spindle is rotating

In some cases, protection may also be needed for head and ears, perhaps even nose

Never remove cutting chips by hand, with or without gloves on

Do not use rags or gloves around moving or rotating objects

When lifting heavy objects, ask for help, use a crane or do not lift

Machine Environment Safety

Make sure the floor is swept, free from oil, water, chips, and other hazards

Check the walkways, so they are not blocked from any direction

See whether all the material is safely stored and finished parts are in proper containers

Machine Tool Safety

Do not remove guards and protective devices

Read and follow operating manuals

Check fixtures and tools before they are used

On the machine, make sure all the tools are tight in the holders, that the tools are sharp and selected properly for the job on hand

Stop all machine motions when measuring or inspecting finished work

Do not leave objects on top of machines

Use only a suitable coolant mixture, and keep the coolant tank clean at all times

Never use a file for breaking corners or a sandpaper for surface polishing during the program execution

Deburr sharp edges before handling a part

Stop all machine power for maintenance

Do not operate a faulty machine

Do not alter design or functionality of the machines or controls

Electrical or control maintenance should be done by authorized personnel

Do not use a grinding machine near the CNC machine slides

Do not use a welding equipment on CNC machine under power

Behave responsibly - do not engage in pranks and horseplay around machinery

These are only some commonsense suggestions, not a comprehensive list for CNC machining safety.

Always observe the company safety policies, as well as safety laws of a particular jurisdiction

G. SHUTTING DOWN A CNC MACHINE

When the CNC machine is not used for an extended period of time, it should be shut down. Many users assume that shutting down a CNC machine means just to turn the power off. There is more than that to shutting down a machine tool with a power switch.

Emergency Stop Switch

The purpose of emergency switch (E-stop) is to *stop all machine motions immediately*, regardless of the current operational mode. When pressed, the switch will lock in place and must be rotated manually in the opposite direction to be released. It should be used sparingly and only in real emergencies, such as when:

An imminent situation that is unsafe to the human being is about to occur

An imminent collision of the machine tool elements is about to occur

In certain situations, it is possible to cause damage to the machine and tooling when pressing the *Emergency Switch*. Depending on the machine design, there may be several emergency stop switches available, located at convenient places. CNC operator should always know the locations of each emergency stop switch. Emergency switch is also called the *E-stop* or *E-switch*.

W A R N I N G!

Although the emergency stop switch disconnects all power to the machine axes, electrical power is still supplied to the CNC machine

For a complete safety shutdown, always follow proper procedures as enacted by company policies.

When *Emergency Stop* switch is released or unlocked, the machine does not restart automatically. Machine setup conditions and other conditions have to present before the automatic start can be selected. This condition is usually achieved by pressing the *Power On* switch.

Parking Machine Slides

Several chapters have mentioned a comment that a CNC program cannot be executed unless the machine had been zeroed first. Recall that zeroing a CNC machine while the machine slides

are at - or almost at - the machine zero, is impractical and may result in an overtravel. Machine zero return needs about *one inch* minimum (or 25 mm), to be *away* from the machine zero position in each axis. This position is often easier to reach at the *end* of work than at its beginning. A practical CNC machine operator knows that to shut off the machine when all slides are at the machine zero position causes the subsequent start up to take a little more time.

To avoid any potential problems in the future, some programmers make a small program to bring the machine slides into a safe position at the end of work, before the power is turned off. Although the idea is good, the solution to one problem may cause another problem. If the machine slides are 'parked' repeatedly at the *same* position for a lengthy period of time, various dirt deposits will collect *under* the slides, possibly causing staining or even rusting in and around the 'parking' area. A better way is to let the CNC operator do the positioning of the slides manually. It does not take any more time and the slides will never be too long at any single position. All that is needed is a motion of one axis at a time, to a different position every time. Since it is done manually, there is a better chance that the machine position will be always different.

Setting the Control System

Control panel of a CNC unit has many switches set to a certain state at the time of a shut down. Again, variations exist as to what the proper procedure is, but a good CNC operator will leave the control system in such a state that it does minimize a potentially dangerous situation, when used by the next person. Here are only some possibilities to apply before leaving the control system for a break, or a complete shutdown:

Turn down the feed rate override switch to the lowest setting

Turn down the rapid override switch to the lowest setting

Set mode to JOG or HANDLE

Set the handle increment to X1 (minimum)

Set the Single Block switch ON

Set the Optional Block switch ON

Set operation mode to MDI

If available, remove the Edit key from the lock

Several other precautions could also be used, but the ones listed are the most typical and should ensure reasonable safety precautions.

Turning the Power Off

Procedures vary from one machine to another, so always consult the machine manual first. However, there are some procedures pretty common to all machines. General rule is to *reverse* the procedure of turning the power on. For example, if the procedure to turn the power on is

Main switch ON

Machine switch ON 3. Control switch ON then the power off procedure will be

Control switch OFF

Machine switch OFF

Main switch OFF

Note that in either case, there is not one switch to do all work. This is for the safety of sensitive electronic system of the CNC unit. Also check the exact function of the emergency switch (described earlier), as it relates to the machine shut down procedure.

H. EQUIPMENT MAINTENANCE

To maintain a CNC equipment is a professional discipline of its own. In general, it is better to leave any kind of maintenance to qualified technicians and engineers. CNC machine operator should only be concerned with the basic preventive maintenance, just by taking care of the machine in general. Modern control systems require very little maintenance, usually consisting of some air filter change and similar simple tasks.

Manufacturers of CNC units and machine tool builders supply reference manuals, including special ones for maintenance, with their products. These publications should be a compulsory study reading for any person involved with maintaining machine tools in working order, electrical, electronic, or mechanical. Many machine manufacturers, and even dealers, also offer training courses in maintenance and general troubleshooting.

38. INTERFACING TO DEVICES

A completed CNC program, debugged and optimized for best performance, should be stored for *future use* or *reference*. Before such a program can be stored, it must be first loaded into the CNC memory, tested and optimized. There are many ways of loading a completed part program into the CNC memory. The most basic, and also the most time consuming, method is to simply key-in the program at the machine directly, using the control panel and keyboard. Without a doubt, this is also very inefficient method, prone to errors. It is true, that Fanuc controls offer a feature called *Background Edit*. This is a standard feature on most controls that allows the CNC operator to key-in (and/or edit) one program, while the control runs the machining operations for another program. In practice, however, many operators simply don't take advantage of this feature for various reasons.

In order to load a part program into the CNC memory or unload a program from the CNC memory, a hardware connection called a *data interface* is needed. An interface is usually an electronic device that is designed to communicate with the computer of the CNC unit.

Typical interfaces and storage media are:

Tape reader and tape puncher (obsolete)

Data cassettes (obsolete)

Data cards (obsolete)

Bubble cassettes (obsolete)

Floppy disks (obsolete but still around)

Hard (fixed) disks

Removable devices (flash and USB storage)

ROM (read-only-memory) devices

... and others

Many of these devices are proprietary, many require not only special cabling, but also software drivers that can run these devices. The focus of this chapter will be on the connections that can be easily assembled and those that use standard configurations. There is one industrial standard many of these devices have in common - a standard called an *RS-232C* interface. Well - almost a standard. There is a number of variations that follow the standard in principle but deviate from it to some extent. This digital book is not an in-depth discussion of CNC communications, it only does an overview of the standard as a guideline, not as a solution to all CNC communications.

A. RS-232C INTERFACE

Data transfer between two electronic devices (computers and controls) requires a number of settings that use the same rules for each device. Since each device may be manufactured by a different company, there must be a certain independent standard that all manufacturers adhere to. The RS-232C is such a standard - the letters *RS* stand for '*Recommended Standard*'. Almost every CNC system, a computer, a tape puncher and tape reader, has a connector (known as a *port*) that is marked RS-232C or similar. This port exists in two forms, one with a 25 *pin* configuration, the other with a 25 *socket* configuration. One with the pins is known as the DB-25P connector, one with the socket as DB-25S connector (male/female respectively). *Figure 52-1* illustrates the typical layout.

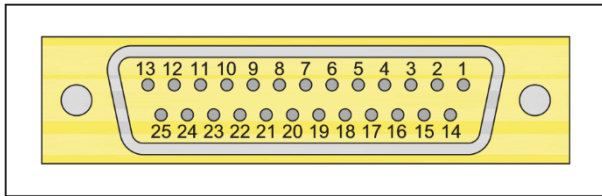


Figure 52-1

Typical 25-pin RS-232C port - DB type

RS-232C port on the CNC unit is usually a standard feature and uses the DB-25S type (the letter *S* means it is a socket type). An external computer, usually a desktop computer or a laptop, together with a suitable cable and communications software is also needed to transfer CNC programs. External devices use mainly the DB-25P type connector (the letter *P* means it is a pin type). The price tag for such a setup (hardware and software) is well below the cost of any suitable alternative, with the exception of flash cards and USB. It is also a very convenient method. CNC program is sent to the system memory and is stored there as long as needed to run the job. CNC operator usually makes some changes and when the job is completed, all changes that are to remain permanent are sent back to the desktop computer or a laptop computer and stored on hard disk. This method works well with a single CNC machine as well as several machines.

Although terms such as *Transmit* (or *Send*) and *Receive* are more common in software, some CNC systems use the terms *Punch* (which is equivalent to *Send*) and *Read* (which is equivalent to *Receive*). These terms go back to the days of punched tape.

To make this very popular method of communications work, only a suitable cable has to be installed between the computer port and the CNC system port. Loading and configuring a communications software that runs the complete operation also has to be done first. In addition, both devices must be set in a way they can 'talk' to each other.

Later in this chapter will be a few notes relating to the basic principles of using a personal computer as an interface with the CNC system. DISTRIBUTED NUMERICAL CONTROL

The Input/Output (I/O) port RS-232C on a CNC machine is used to send and receive data. The external sources are usually a hard disk or a paper tape. In many shops, programs are transferred through the means of DNC, which means *Distributed Numerical Control*. The control has features available to make data transfer possible.

To communicate between one CNC machine and one computer using the RS-232C port, all equipment required is a cable between the two devices and a software. To communicate with two or more machines, using the same single RS-232C port, each machine must be connected to a split box with a cable. Split box is available with two or more outlets, selectable by a switch. This is the simplest form of DNC. It requires well organized procedures to make it work efficiently. DNC is not a part of the control unit and is not covered here. Commercial DNC packages are available at various levels of sophistication and cost.

Some DNC software also allows a useful feature called '*drip-feeding*', which is a method used when the program is too large to fit into the CNC memory.

B. TERMINOLOGY OF COMMUNICATIONS

Communications have their own terminology. There are many terms, but five terms are commonly used in CNC:

Baud Rate

Parity

Data Bits

Start Bit

Stop Bit

Baud Rate

Baud rate is the data transmission speed. It is measured as the amount of data bits per second, written as *bps*. Baud rates are only available in fixed amounts. Typical rates for older Fanuc controls are 50, 100, 110, 200, 300, 600, 1200, 2400, 4800 and 9600 bps. Modern controls can have the baud rate set to 2400, 4800, 9600, 19200, 38400, 57600 and 76800 bps. In terms of time, the higher the rate, the faster the transmission. Single data bit transfer rate will be the result of one divided by the baud rate:

$$S_b = \frac{1}{B}$$

+ where ...

S_b = Time required to transfer a single bit in seconds B = Baud rate in seconds

A single bit transferred at 300 bps will take 0.03333 of a second, but a single bit transferred at 2400 bps will take only 0.00042 of a second. In practice, it takes about 10 bits to transfer one character (see *Stop Bits* section below), so at 2400 bps setting, the transmission will be at a rate of about 240 cps (characters per second). 4800 bps is a good setting once everything is working well. Higher settings are necessary for 'drip-feed' methods.

Parity

Parity is a method of checking that all transmitted data were sent correctly. Just imagine what would happen if some characters or digits of a CNC program were not transferred correctly or not transferred at all. Parity can be *even*, *odd*, or *none*, and *even* is the most common selection for CNC communications. This is similar to punched tape parity.

Data Bits

A *bit* is an acronym for **Binary digit** and is the smallest unit that can store information in a computer. Each binary digit can have a value of either one (1) or zero (0). One and zero represent the ON and OFF status respectively, so a *bit* is something like a toggle switch that can be turned on and off as needed. In the computer, every letter, digit, and symbol used in a CNC program is represented by a series of bits, eight bits to be precise, that create a unit called a *byte*.

Start and Stop Bits

To prevent loss of data during communication, each byte is preceded by a special bit called the *start bit*, which is low in voltage level signal. This signal is sent to the data receiving device and informs it that a byte of data is coming next.

A bit similar to the start bit, but at the *end* of the byte, has exactly the opposite meaning. It sends a signal to the receiving device that the byte has ended or stopped being transmitted. This bit at the end of a byte is called the *stop bit*. Because the start and stop bits go together, they are often teamed up together as the stop bits and set the devices to *two stop bits*.

Many terms exist in communications. With growing interest, this is a very rich field to study.

C. DATA SETTING

Data used for communications must be set properly before data transfer can begin. The setting at one end (computer or the CNC system) must match the setting at the other end. For baud rate, consult the machine manual - a good start is at 2400 bps. Newer models have a higher default. Typical software setting is done through the configuration at the computer end and through the CNC system parameters at the CNC end. Settings at both ends must match. Typical Fanuc settings

are:

4800 bps baud rate

Even parity

7 data bits (seven data bits)

2 stop bits (two stop bits)

Proper connection depends mainly on the configuration of connecting data cables.

D. CONNECTING CABLES

The most common cable for communication between a CNC machine and a computer is a shielded and grounded cable, containing several small wires (at least eight), each one enclosed in a colored plastic sleeve. Main purpose of making a communication cable is to connect the CNC port (usually 25 sockets) with the computer port (usually 25 pins), using a properly configured cable. Always use a cable of high quality. Shielded cables can reach farther distances and are generally better choice to withstand interferences during data transmission. Wires are identified by their gauge value, for example a 22-gauge or a 24-gauge wire is a good choice for communications.

The 25-pin port has each pin or socket numbered (see the first page of this chapter) and individual wires of the cable have to be connected to proper numbers at each end. It is quite common to 'cross' the wires between each end. Typical crossing would be between the pin number 2 and the socket number 3, and a pin 3 and socket 2. Some numbered positions have to be connected *at the same end* of the cable. This is called 'jumping'.

Null Modem

A very common cable wiring that is used in general communications is called a *null modem*. Connection of the two ends follows a certain standard, shown in *Figure 52-5*. Each number represents the pin or socket on the DB-25 connector. Note the jumps between connections 6 and 8 at both ends. *Figure 52-6* shows the same null modem configuration in a graphic way. This is a very popular method showing cable configurations.

PIN DB-25P	SOCKET DB-25S
1	1
2	3
3	2
4	5
5	4
7	7
6 and 8	20
20	6 and 8

Figure 52-5

Null modem pin connections

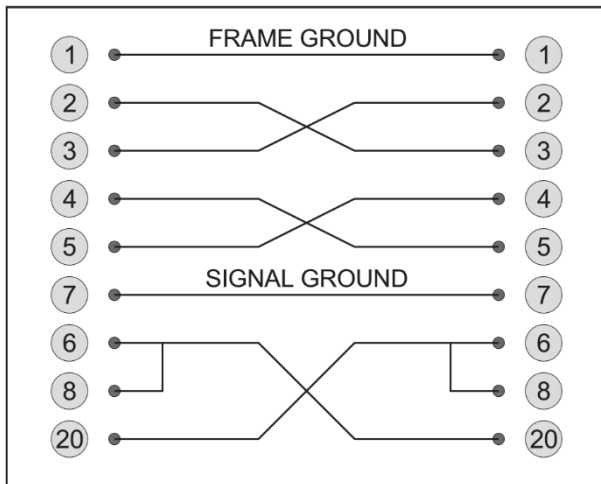


Figure 52-6

Graphic representation of null modem connections

Cabling for Fanuc and PC

As the most common cable communication will be between a Fanuc control and a desktop computer or a laptop, *Figure 52-7* illustrates a typical cable configuration. Note the similarity to the null modem configuration.

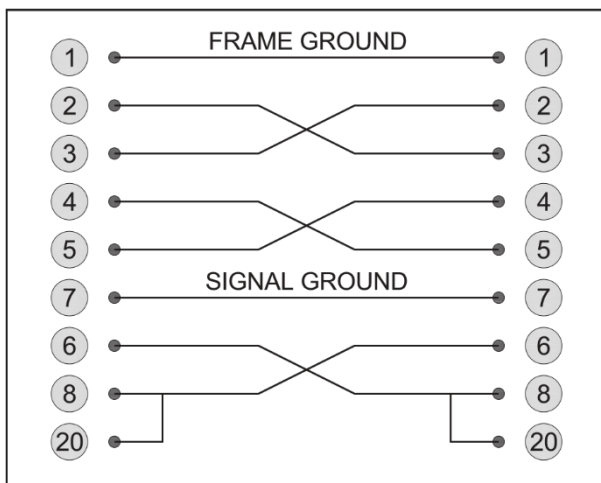


Figure 52-7

Typical cable configuration for Fanuc controls

Regardless of what cable configuration will be used, a good communication software that will run the whole operation is also needed. Some companies use a software specially designed for CNC work, others purchase very inexpensive communications software, sold by the majority of computer stores.

39. MATH IN CNC PROGRAMMING

Math in programming - the single word 'math' often appears to be so powerful that it strikes a weak chord in many programmers. It is surprising how many new programmers, manual programmers in particular, are afraid of the often numerous calculations associated with CNC programming. This fear is really not substantiated. Let's look very briefly at what kind of mathematical knowledge is really necessary to handle typical programming calculations for manual program preparation.

First, the basic arithmetic functions - *addition*, *subtraction*, *multiplication* and *division* - are at the core of any mathematical activity. Going a bit further, the knowledge of common algebraic functions is definitely useful, mainly *square roots* and *powers of a number*.

Second, since CNC programming is based on the relationship of points within a system of rectangular or polar coordinates, a good knowledge of *basic geometry* is also imperative. The scope of this knowledge should cover understanding many principles of *angles*, the concept of *degrees* and their subsets, *tapers*, *polygons*, properties of an arc and a circle, the *pi* constant (π), and other associated topics. Knowledge of *planes* and *axial orientations* is important in many cases as well.

Without a doubt, the most important part of geometry, one that absolutely *must* be mastered, is the solution of *right angle triangles*, using *trigonometric functions*. Very seldom there will be a problem or calculation that will require a solution using oblique triangles, although these problems may arise.

Knowledge of trigonometry is essential to any serious CNC programming

Most difficulties in solving trigonometric problems are not as much in the ability to use a specific formula and solve the triangle - but in the *inability to see* the triangle to be solved in the first place. Often, programming involves a drawing that is very complex in terms of part geometrical definitions. Such a drawing will have so many elements, that overlooking the obvious is possible, even likely.

Any specific knowledge of analytic and spatial geometry is not really required for a 2 and 2-1/2 axis work, but it is essential for a work in all three axes, particularly for complex surfaces, 3D tool path and multi surface machining or surface manipulation. However, this kind of programming is not done without a computer and CAD/CAM software.

There are several specific mathematical subjects to learn and to know in depth. All of them have been selected only for their importance in CNC programming and are described here in the necessary detail.

E. BASIC ELEMENTS

Arithmetic and Algebra

The subject of *arithmetic* deals with handling numbers involving the four basic operations:

Addition

Subtraction

Multiplication Division

Algebra is an extension of arithmetic and deals with handling numbers in terms of equations and formulas. Typical usage will involve:

Square roots

Powers of a number

Trigonometric functions

Solving formulas and equations

Variable data

In algebra, typical work involves several known values and one or two unknown values. Using various formulas and equations, unknown values can be solved (calculated) to achieve the desired result.

Order of Calculations

In the field of mathematics, there is a precisely defined order in which the calculations are performed. Every electronic calculator is based on these centuries old rules. In a combination of various algebraic operations, the order of calculations will follow these rules:

Multiplications and divisions are always calculated first

Additions and subtractions follow, order is not important

Any roots, powers to a number, and operations within parentheses are always calculated before multiplications and divisions.

The following calculation will have the same result with or without parentheses:

$$3 + (8 \cdot 2) = 19$$

$$3 + 8 \cdot 2 = 19$$

The multiplication is always performed first, regardless of whether it is enclosed in parentheses or not. If addition must be done first, it must be enclosed within parentheses:

$$(3 + 8) \cdot 2 = 11 \cdot 2 = 22$$

These two examples show that even an innocently looking small omission may have significant consequences.

F. GEOMETRY

For all practical purposes in manual CNC programming, there are only three entities in the engineering drawing:

Points

Lines

Circles and Arcs

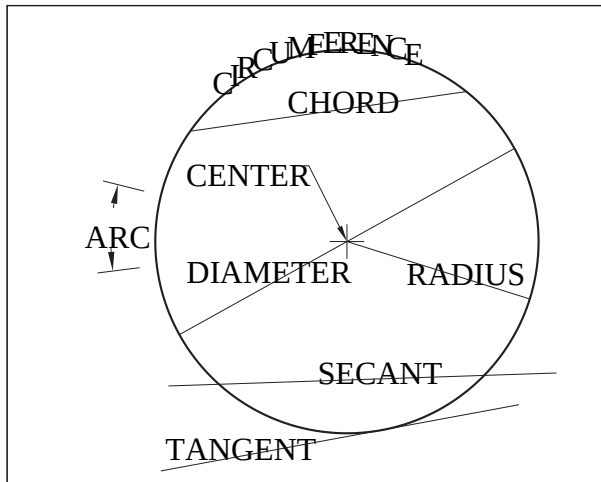
Points have no parts and are represented by the XY coordinates in a 2D plane or by XYZ coordinates in 3D space. Points are also created by an intersection of two lines, two circles or arcs, and a line and a circle or arc.

Point is also created by a line tangent to a circle or an arc, a circle or an arc tangent to another circle or an arc.

Lines are straight connections between two points creating the shortest distance between the points.

Circles and *Arcs* are curved elements that have at least a center and a radius.

Other elements such as *splines* and *surfaces* are too complex for manual programming, although they are also based on the same fundamental elements.



Circle

Circle is mathematical curve, where every point on the curve has the same distance from a fixed point. This fixed point is called a *center*.

Several terms are directly related to a circle - *Figure 53-1*:

CENTER - is a point from which a circle or an arc is drawn with a given radius

RADIUS (radii or radiuses in plural) - is a line from the center to any point on the circumference of the circle

DIAMETER - is a line through the center between two points on the circumference of the circle

CHORD - is a straight line joining any two points on the circumference of the circle

ARC - is any part of the circle between two points on the circumference of the circle

CIRCUMFERENCE - is the length of the circle

(Length of the curve that bounds a circle)

TANGENT - is a point where a line, an arc or another circle touches the circumference of the circle but does not cross it - this point is known as the point of tangency

SECANT - is a straight line that passes through a circle and divides it into two sections

Two area sections of a circle have their own names. They are called the *sector* and the *segment* of a circle, and are shown in *Figure 53-2*:

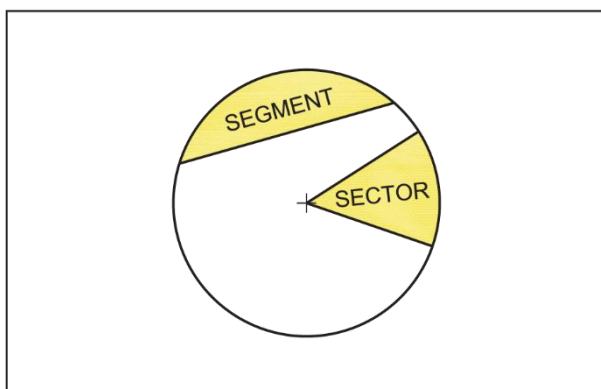


Figure 53-1

Basic elements of a circle

Figure 53-2

Segment and sector of a circle

SECTOR - is an area within a circle formed by two radii and the arc they intercept

SEGMENT - is an area within a circle formed by the chord and its arc

Neither the sector nor the segment of a circle play any significant role in CNC programming.

PI Constant

PI is a Greek letter used in mathematics to represent the ratio of the circle circumference to the circle diameter. Its symbol is, it is pronounced 'pie', and has the value of 3.141592654..., and regardless of how many decimal places will be used, it will always represent only an approximate value. For programming purposes, use the value returned by a calculator or computer, usually with six to nine decimal places. In both cases, the internal value is a lot more accurate than the displayed value. In many cases, the rounded value of 3.14 is sufficient for most results.

Circumference of a Circle

The length of a circle - or its *circumference* - is seldom needed for programming and is included here only to enrich the general theory. It can be calculated from the following formula using the *pi* constant:

$$C = 2 \pi r$$

or

$$C = \pi D$$

+ where ...

C	=	Circle circumference
π	=	Constant 3.141592654...
r	=	Circle radius
D	=	Circle diameter

Length of Arc

The length of an arc is also a rare requirement and can be calculated from the following formula:

$L = \frac{C}{360} \times A$

+ where ...

C	=	Circle circumference
π	=	Constant 3.141592654...
r	=	Circle radius
A	=	Arc angle

There are two other very important calculations relating to a circle. They are used in programming very often and should be understood well. One is based on the *chord* of a circle, the other on the *tangency* of a circle. As both calculations require the knowledge of trigonometry, they will be described later in the chapter (see page 508).

Quadrants

Quadrant - is the part of a circle formed by the system of rectangular coordinates (described on page 15), where the axes pass through the center of the circle. There are four equal quadrants in a circle, identified by Roman numerals I, II, III and IV, starting at the upper right quadrant along the counterclockwise direction - Figure 53-3.

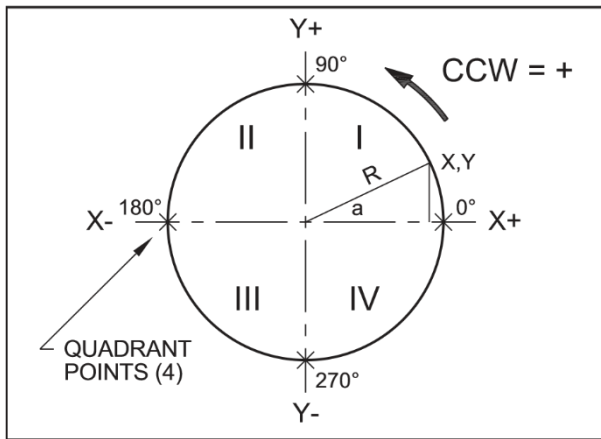


Figure 53-3

Quadrants of a circle and the mathematical definition of angular direction

Each quadrant is exactly 90, crossing at circle quadrant points. Therefore, a circle has the sum of all four angles equal to 360. Angles are always counted counterclockwise as positive, starting from zero degrees (0).

Quadrant points (also known as *cardinal points*) are often compared to a hand direction on the face of an analogue clock or as a direction of a compass pointer. 0° is arbitrarily located at the equivalent position of *three o'clock* or *East* direction, 90 at *twelve o'clock* or *North* direction, 180 at *nine o'clock* or *West* direction, and 270 at *six o'clock* or *South* direction - Figure 53-4.

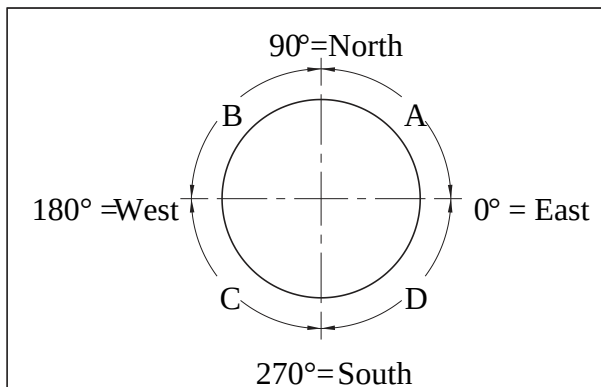


Figure 53-4

Angles and quadrants - 0 is East direction or 3 o'clock direction on the face of a standard analogue clock

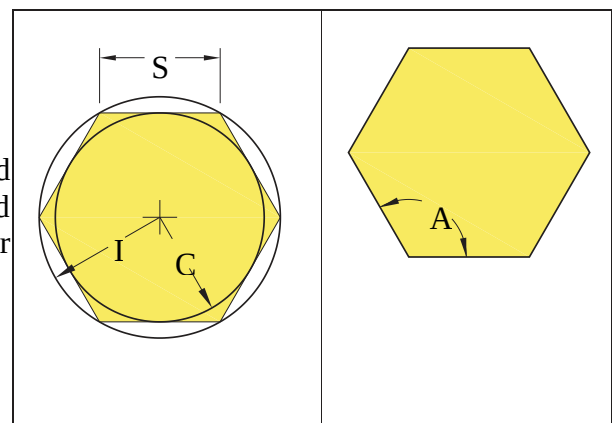
G. POLYGONS

Polygon is a common geometric element defined by a number of straight line segments that are joined at the end points. These line segments are the *sides* or *edges* of the polygon - Figure 53-5.

Figure 53-6

Regular polygon

Inscribed and circumscribed circles and a single angle



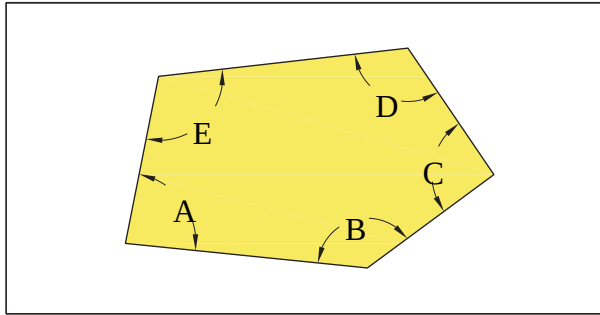


Figure 53-5

Sum of angles in a polygon

The sum of all angles in a polygon can be calculated from the following formula:

$$S = (N - 2) \times 180$$

+ where ...

S = Sum of the angles

N = Number of sides in the polygon

For example, a five sided polygon shown in the illustration has the total sum of angles:

$$S = (5 - 2) \times 180$$

$$S = 540$$

There are several different polygons used in geometry, but only *one special kind* is of interest to CNC programming. This polygon is called a *regular polygon*, all others are irregular polygons. Regular polygon is a polygon where all sides are of equal length, called *equilateral sides*, and where all angles are also equal, called *equilateral angles* - Figure 53-6.

A single angle in a regular polygon can be calculated from this formula:

$$A = \frac{(N - 2) \times 180}{N}$$

+ where ...

A = Single angle in degrees

N = Number of sides in the polygon

For example, a six sided polygon (commonly known as the *hexagon*) has a single angle of 120:

$$A = \frac{(6 - 2) \times 180}{6}$$

$$A = 120$$

A regular polygon is quite often defined by the number of its sides and its center, located within an *inscribed* or *circumscribed* circle. Figure 53-6 above illustrates the concept of inscribed and circumscribed polygon, as it applies to a hexagon.

Although regular polygons may have virtually unlimited number of sides, some polygons are so common that they have a special descriptive mathematical name:

Number of sides	Common name
3	Triangle

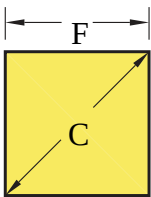
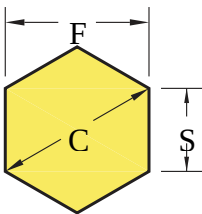
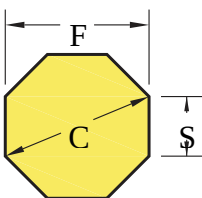
4	Square (Quadrilateral)
5	Pentagon
6	Hexagon
7	Heptagon
8	Octagon
9	Nonagon
10	Decagon
12	Dodecagon
n	n -gon
	$C = F \times \sqrt{2}$
	$F = C \times \sin 45^\circ$
	$C = F / \cos 30^\circ$ $C = 2 \times S$
	$F = C \times \cos 30^\circ \quad F = S / \tan 30^\circ$
	$S = F \times \tan 30^\circ$ $S = C / 2$
	$C = F / \cos 22.5^\circ \quad C = S / \sin 22.5^\circ$
	$F = C \times \cos 22.5^\circ \quad F = S / \tan 22.5^\circ$
	$S = F \times \tan 22.5^\circ$ $S = C \times \sin 22.5^\circ$

Figure 53-7

The most common regular polygons - square, hexagon and octagon

In Figure 53-7 are three most common regular polygons a *square*, a *hexagon* and an *octagon*. Calculations of the distance between opposite corners C , the distance between flats F and the length of each side S are given. Note that a hexagon may have two different orientations (two horizontal sides or two vertical sides), which have no effect on the calculations. Hexagon orientation can be compared in Figure 53-6 with the hexagon orientation in Figure 53-7.

H. TAPERS

All taper calculations are virtually confined to the lathe machining exclusively. Infrequently, tapers also appear in milling applications. All tapers in this section relate to the lathe applications (so called *circular tapers*) but can be modified to milling. The main purpose of tapers is to provide

a match between assembled parts. By definition,

Taper is a uniformly created conical surface on a pin or in a hole

Many tapers are industry standards and are used for small tool holders (shanks), such as a Morse taper or a Brown and Sharpe taper. In addition, there standard tapered pins, machine spindle tapers, tool holder tapers, etc. In most cases, the taper is normally defined by the large end diameter, its length and a special note describing the taper.

The description varies between imperial and metric standards. For example, AMER NATL STD TAPER NO. 2

(American National Standard Taper number 2) is a specific taper description. Another common description in imperial units is a *taper per foot*. Metric system is much simpler, using only a ratio. Ratio is used in imperial drawings as well. In both measuring systems, there is one common rule:

Taper on diameter is the difference in diameter per unit of length

Taper Definition

Most drawings define a taper in two common ways:

One diameter and length with taper description or note

Diameter at both ends and the length with taper description or note

If a single diameter is defined, it is often the larger one.

The taper description is a note with an arrow pointing towards the taper. In imperial measurements, the note may identify a standard taper or a taper per foot (TPF). In metric, the taper is always a ratio. *Figures 53-8 and 53-9* show the differences between the two units, which is confined to the taper identification.

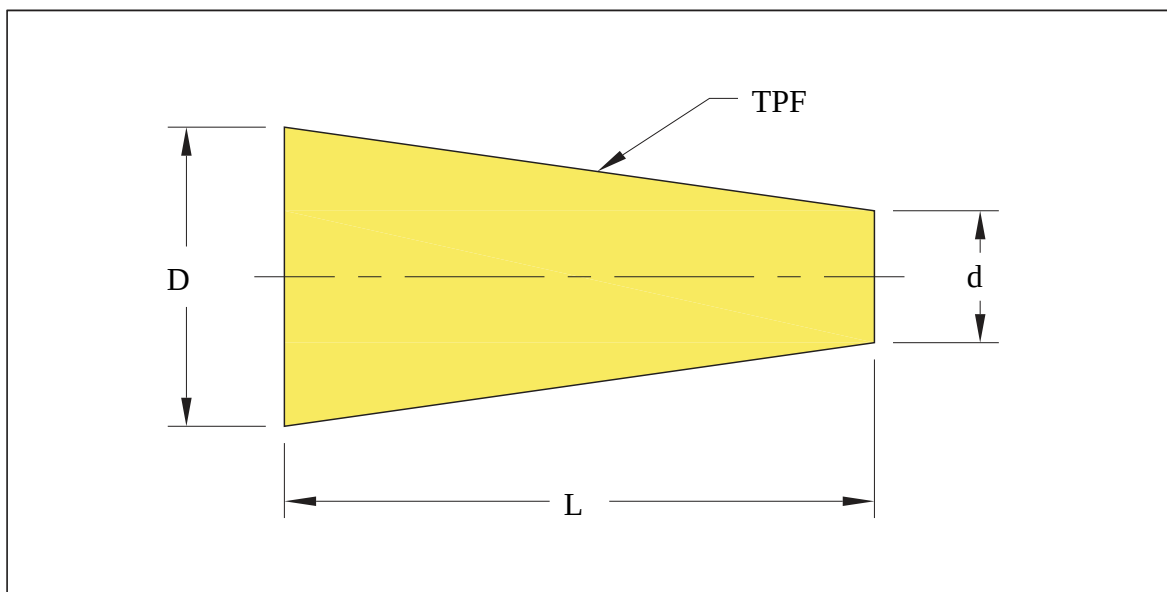


Figure53-8

Circular taper-Imperial description

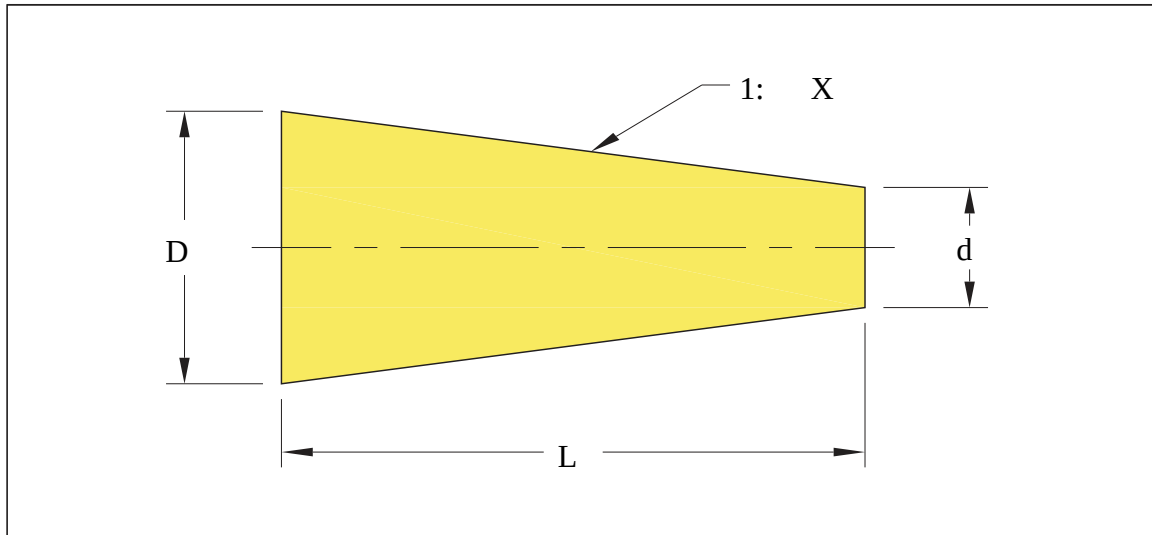


Figure 53-9

Circular taper - Metric description

In the *Figure 53-8*, showing imperial units method, the letters have the following meaning:

+ Dimensions ...

D = Diameter at the large
end in inches

d = Diameter at the small
end in inches

L = Length of taper in inches

TPF = Taper per foot in inches

X = Ratio value 1: X (not
shown)

In the *Figure 53-9*, showing metric units method, the letters have the following meaning:

+ Dimensions ...

D = Diameter at the large end in
millimeters

d = Diameter at the small end in
millimeters

L = Length of taper in
millimeters

X = Ratio value 1: X

All formulas in this section use these designations.

Taper Per Foot

Taper per foot is defined as:

Taper per foot is the difference on diameter in inches over one foot of length

For example, a taper defined as 3.000 inches per foot, abbreviated as 3.0 TPF or 3 TPF in the drawing, is a taper that will change the conical diameter by 3 inches for every 1 foot of length.

Taper Ratio

Metric definition of a taper is similar:

Taper is defined as the ratio of difference between large diameter and small diameter over a given length of the cone

The metric specification of a taper is the ratio X:

$$\frac{1}{X} = \frac{D - d}{L}$$

The ratio 1: X means that over the length of X mm, the diameter of the cone will change (either as an increase or as a decrease) by 1 mm.

For example, a taper specified as 1: 5 will increase 1 mm on diameter, every 5 mm of length.

For milling, the taper is defined as the difference in width over a given length (per side).

Taper Calculations - Imperial Units

Missing drawing dimensions in *Figure 53-8* may be calculated from the given data. If the taper ratio is not specified (the normal case), but we want to know what the ratio is, the following formula will help. To calculate the taper ratio amount X, when D, d and L are known:

$$X = \frac{L}{D - d}$$

To calculate the small diameter d, with D, L and TPF:

$$d = D - \frac{L}{\text{TPF}}$$

To calculate the large diameter D, with d, L and TPF:

$$D = \frac{L}{\text{TPF}} + d$$

To calculate the length L, if D, d, and TPF are known:

$$L = \frac{(D - d) \times \text{TPF}}{12}$$

Taper Calculations - Metric Units

Missing drawing dimensions in *Figure 53-9* may be calculated from the given data. In metric system, the taper ratio is normally known, other dimensions can be calculated.

To calculate the small diameter d, with D, L and X:

$$d = D - \frac{L}{X}$$

To calculate the large diameter D, with d, L and X:

$$D = d + \frac{L}{X}$$

X

To calculate the length L, if D, d, and X are known:

L (D d) X

To calculate the ratio X (if unknown), with d, D and L:

X — L
D d

I. CALCULATIONS OF TRIANGLES

The most common geometrical entity in programming is a triangle. All triangles are polygons, but not all triangles are regular polygons. All triangles have three sides, although not always of the same length. There is a number of different triangles in geometry, but only a handful are used in everyday CNC programming.

Types of Angles and Triangles

Three main groups of triangles can be grouped together by their angles - *Figure 53-10*.

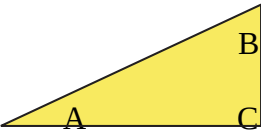
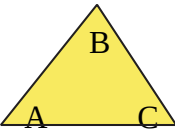
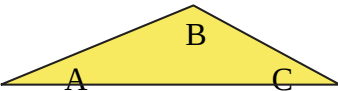
a		$< 90^\circ$ $< 90^\circ$ $\angle C = 90^\circ$
b		$< 90^\circ$ $< 90^\circ$ $< 90^\circ$
c		$< 90^\circ$ $> 90^\circ$ $< 90^\circ$

Figure 53-10

Typical triangles a) Right triangle b) Acute triangle c) Obtuse triangle

Some more detailed definitions may be useful:

RIGHT angle means that the given angle is equal to 90

ACUTE angle means that the given angle is greater than 0 and smaller than 90

OBTUSE angle means that the given angle is greater than 90 and smaller than 180

A right triangle is also called a right angle triangle

It defines a triangle that has one right angle (90°)

An acute triangle is also called an acute angle triangle

It defines a triangle that has three acute angles

An obtuse triangle is also called an obtuse angle triangle It defines a triangle that has one

obtuse angle

In addition, there is also an *oblique* angle, which is not a new type of an angle, just a new definition:

OBLIQUE angle can be either an acute or an obtuse angle, which means it cannot be 90 or 180

All triangles share a single feature - the *sum* of all angles in a given triangle is always equal to 180 - *Figure 53-11*.

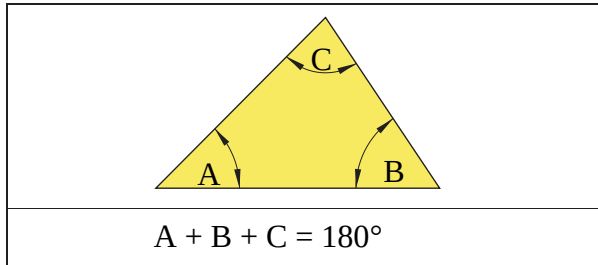


Figure 53-11

Sum of all angles in a triangle is always 180 degrees

The *oblique* triangle - and its close cousin the *isosceles* triangle - are types of triangles seldom ever needed in programming. However unlikely, it is always possible. These triangles can be solved only if at least three dimensions are known, and one of them must always be a side:

One side and two angles must be known

Two sides and the angle opposite one of them

Two sides and the included angle

Three sides

Isosceles triangle has two sides of equal length. Each side - or leg - is joined by a line called the base. The two angles at the base are always equal - *Figure 53-12*.

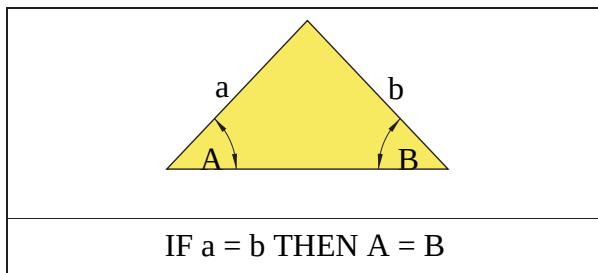
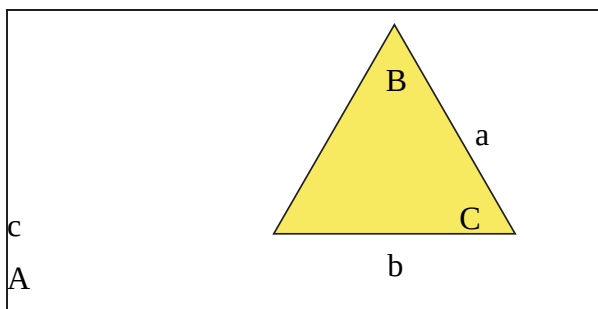


Figure 53-12

Isosceles triangle

A triangle that has all sides of equal length is called an *equilateral* triangle. An equilateral triangle is also always an *equiangular* triangle, because all internal angles are the same - each angle is 60 - *Figure 53-13*.



$a = b = c$	$A = B = C = 60^\circ$

Figure 53-13

Equilateral triangle

Right Triangles

A *right triangle* - or a *right angle triangle* - is triangle that has one angle equal to 90 (a triangle with two or more right angles is impossible). As there are 180 in any triangle (sum of all angles), that means the sum of the two remaining angles must also be 90. There is a number of mathematical relationships that form the base of all calculations. Here is a look at those that are important in CNC programming. Learn these relationships well enough to be able to apply them to daily situations. Keep in mind that 99.9% of all triangles to be solved are right triangles.

The side of a right triangle that is opposite the right angle is called the *hypotenuse* and is always the longest side of the triangle. The other two sides are called *legs*. The illustration in Figure 53-14 shows a right triangle, where C angle is the right angle (90°), and the side c is the hypotenuse. The sides opposite to angles have a low case identification corresponding to the angles described in capital letters.

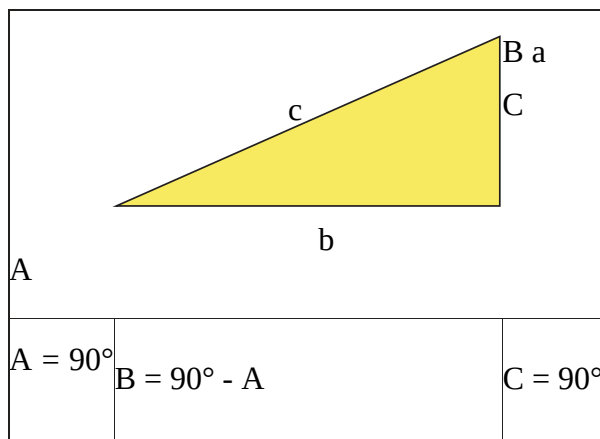


Figure 53-14

Right angle triangle and the relationship of angles

A circle drawn inside of a right triangle that is tangent to all three sides a , b , c - Figure 53-15 - has a diameter D calculated from this formula:

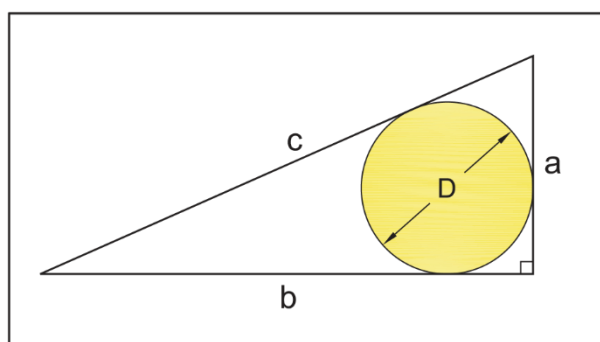


Figure 53-15

Circle inscribed in a right triangle

D a b c

An inscribed angle in a semicircle is always 90, as shown in Figure 53-16. Line AB is the circle diameter D .

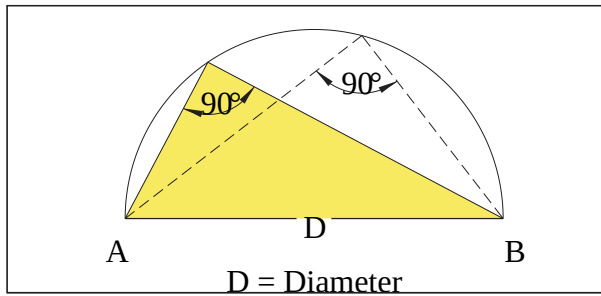


Figure 53-17

Inscribed angle in a semi-circle

In Figure 53-17 is a line from point A to the center of circle B. A line from point A to the tangency of the circle will create either a point C or point D. The angle a is created between lines AC and AD, where the line AB is a *bisector* of the angle a , creating two equal angles. The two angles a_1 and a_2 as well as triangles ABC and ABD are identical.

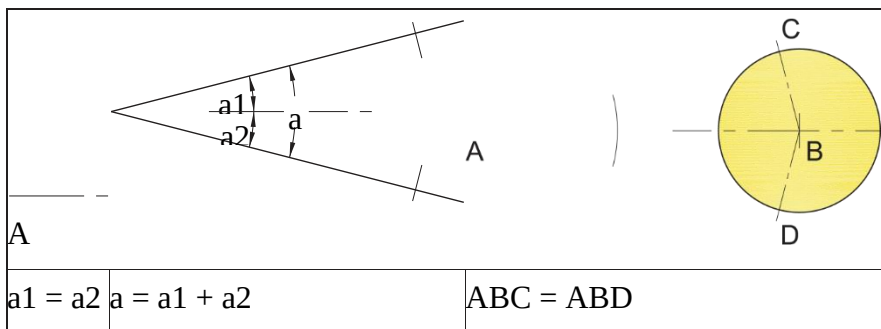


Figure 53-17 Bisector creates two equal angles

Similar Triangles

Triangles are considered similar if they have their corresponding angles *equal* and their corresponding sides *proportional*. Any two triangles are similar, if:

Two angles of one triangle are the same as two angles of the other triangle

An angle of one triangle is the same as the angle of the other triangle and the including sides are proportional

Both triangles are similar to another triangle

The corresponding sides of the two triangles are proportional

In CNC programming, mathematical relationship of two triangles are used frequently - for example, when machining tapers or similar angular items. A good example is a taper specified in the drawing that must be extended at one or both ends, to allow for any necessary tool clearances.

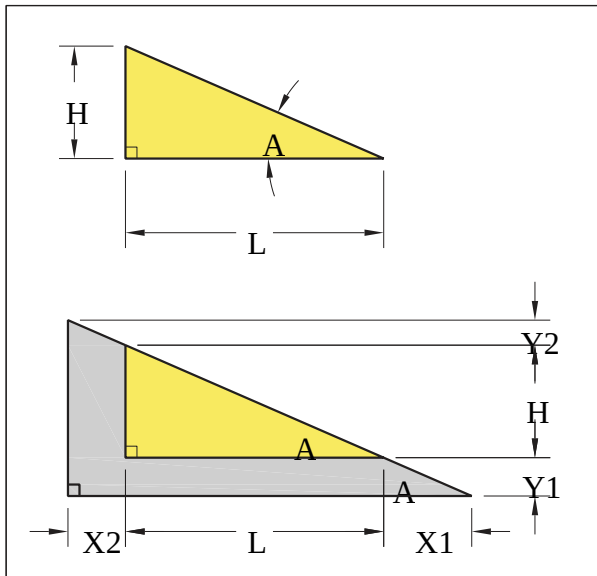


Figure 53-

Similar triangles - 1

The illustration in Figure 53-18 shows the relationship between two *similar* triangles. The same illustration also includes several dimensions important for programming:

+ where ...

L Original length
=

H Original height
=

A Common (shared)
= angle

X1 Front clearance in
= X axis

X2 Back clearance in
= X axis

Y1 Front clearance in
= Y axis

Y2 Back clearance in
= Y axis

Figure 53-19 shows the same two triangles in a simplified way. In the upper part, the values X and Y are sums of the extensions (clearances) from the previous example:

$$X = X1 + X2$$

$$Y = Y1 + Y2$$

The illustration bottom part shows the relationship of the opposite sides H and U to the adjacent sides L and W .

The relationship of sides as a formula is:

$\frac{H}{L} = \frac{U}{W}$

U	W

If *three* values are known instead of only two, the unknown value can be calculated using yet another formula. For example - the values *H*, *L* and *W* are known, while the *U* value has to be calculated, using these sample values - *H* is 13 mm, *L* is 45 mm and *W* is 57 mm. To calculate side *U*, the previous formula has to be reversed:

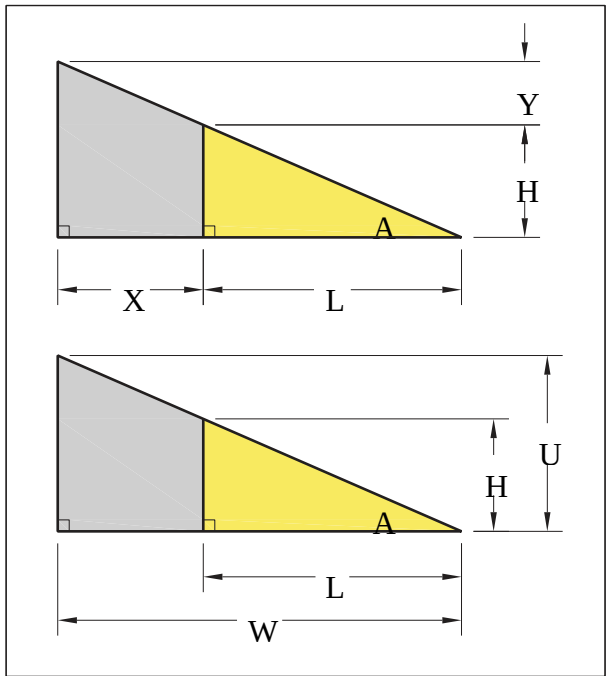


Figure 53-20
Similar triangles - 2

U	W	H
L		

With known values entered, the *U* side can be calculated. If *U* is isolated on the left and the known values on the right of the equation, the calculation is simple:

$$U = (57 \cdot 13) / 45 \quad U = 16.466667 \text{ mm} \sim 16.467 \text{ mm}$$

Sine - Cosine - Tangent

Figure 53-20 shows the most important relationships of sides and angles of a right triangle.

$a \sin A = \frac{a}{c}$	$\cos A = \frac{b}{c}$	$\tan A = \frac{a}{b}$
$b \sin B = \frac{b}{c}$	$\cos B = \frac{a}{c}$	$\tan B = \frac{b}{a}$

Figure 53-20

Trigonometric functions - sine, cosine, and tangent

This relationship has its own terminology and is defined as a **ratio of sides**, using the *sine*, *cosine* and *tangent* functions of the given angle. Other available functions, namely *cotangent*, *secant*, and *cosecant*, are normally not used in CNC programming.

Sine of an angle - abbreviated as *sin*- is a ratio of side opposite the angle to the hypotenuse of the triangle

Cosine of an angle - abbreviated as *cos*- is a ratio of side adjacent to the angle to the hypotenuse of the triangle

Tangent of an angle - abbreviated as *tan*- is a ratio of side opposite the acute angle to the side adjacent

Inverse Trigonometric Functions

From all definitions, the value of sine, cosine and tangent is always expressed as a *ratio of two sides*. The angle that can be calculated from this ratio is the result of an *inverse trigonometric function*. An inverse function is sometimes associated with the word *arc*, as a prefix to any standard function. For example, *arc sin* of an angle *A* is the *angle whose value* is the ratio of the side *a* and the hypotenuse *c*.

With most pocket calculators, the inverse function can be selected by pressing the **2nd** key + the standard function (*SIN*, *COS*, *TAN*) - often shown as the *power of minus 1*. Actual keystrokes vary from one calculator to another, but the result is always the same:

If ...	Sin A = a / c
Then ...	A = arc sin (a / c)
Or...	A = sin⁻¹(a / c)
If ...	Cos A = b / c
Then ...	A = arc cos (b / c)
Or...	A = cos⁻¹(b / c)
If ...	tanA = a / b
Then ...	A = arctan(a / b)
Or...	A = tan⁻¹(a / b)

While there is only a *single* result for any trigonometric function, there could be several results for the inverse function. For example, value of 0.707106781 is the SIN of 45, as well as the SIN of 135, and COS of 45.

Degrees and Decimal Degrees

Less common method of angle calculations in programming is *conversion of angles*. This method applies only to drawings using *minutes* and *seconds* that define precision of angles, rather than the more accurate *decimal degrees*. Older designations abbreviated as *DMS*, or *D-M-S* (for *degrees-minutes-seconds*) dimensions are not suitable for program development and have to be converted to decimal degrees. Decimal degrees are **always** necessary for calculations of coordinate points, and the conversion is required.

A simple formula can be used to convert *degrees-minutes-seconds* to *decimal degrees*:

DD	D	M	_____	S
60	3600			

+ where...

DD Decimal degrees

=

D Degrees (as
= ' drawing)

M Minutes (as
= ' drawing)

S Seconds (as
= ' drawing)

Therefore,

6448 ' 27 " ...is equivalent to (parentheses are not required):

$$\mathbf{64 + (48 / 60) + (27 / 3600) = 64.8075^\circ}$$

Abbreviations *DMS* or *D-M-S* and *DD* or *D-D* are common features of all scientific calculators. Interesting, but much less useful, is conversion to change *decimal degrees* to *DMS* (*DD*=>*DMS*). Such a conversion is not required in CNC programming, with the exception to perform a double check - to verify that the original converted result is correct. Calculation of *DD* to *DMS* is nothing more than isolating the decimal part of *DD* into three steps. For example, to convert 29.545021° to degrees-minutes-seconds (DMS) format, these three steps are required:

[1] The first step is to isolate the whole degrees amount from the decimal degrees:

$$\mathbf{29.545021 - 0.545021 = 29}$$

[2] The seconds step is to take the decimal portion and multiply it by sixty, to get the minutes:

$$\mathbf{0.545021 \ 60 = 31.701126 = 32'}$$

[3] The third and final step is to find seconds by taking the last decimal part and multiply it by sixty to get seconds:

$$\mathbf{0.701126 \ 60 = 42''}$$

The final DMS value of the example will be 2932'42", with an insignificant rounding error.

Pythagorean Theorem

The key work of the Greek mathematician Pythagoras (6th century B.C.) is known today as the *Pythagorean Theorem* - it is generally covered early in a high school mathematics curriculum. This theorem has become one of the core concepts of trigonometry - it states:

In any right triangle, the square of the hypotenuse is equal to the sum of squares of the other two sides

Pythagorean Theorem is used in CNC programming to find the length of *any* side in a right triangle, providing the *two other sides are known*. *Figure 53-21* shows calculation of each side of a right triangle, based on two known sides:

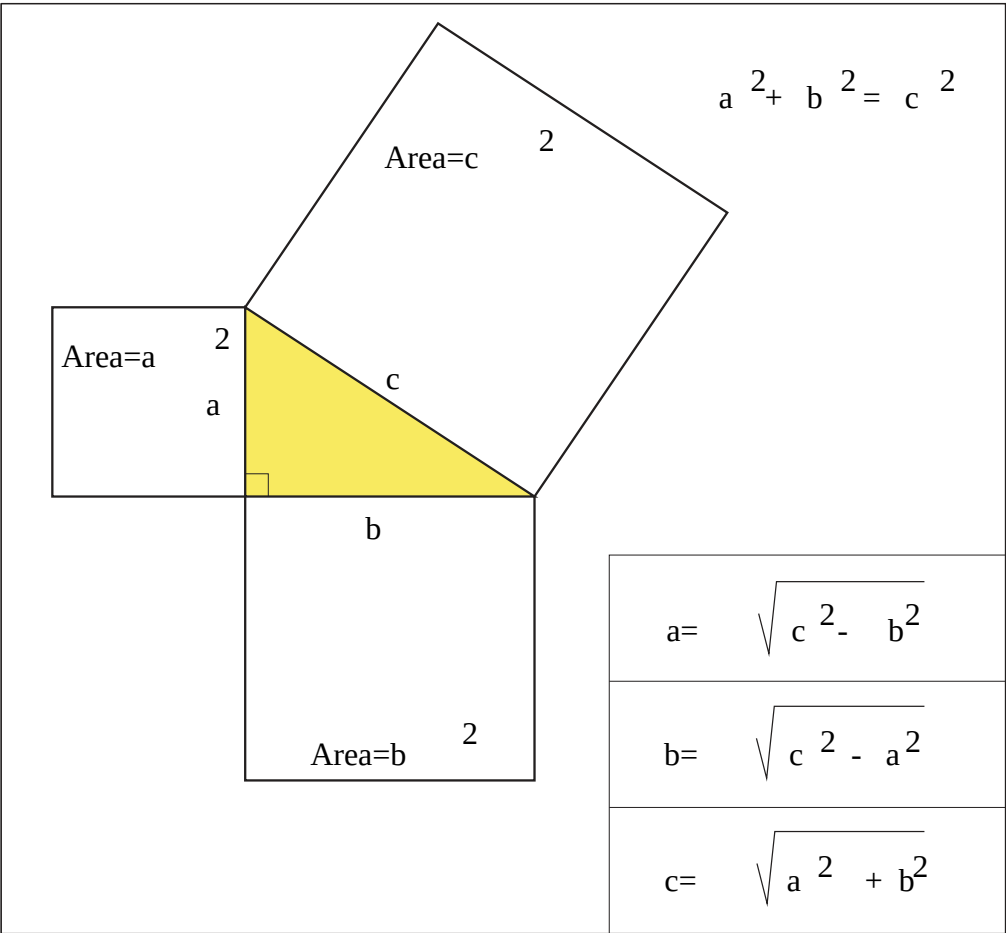


Figure53-21

Example ...

If the length of hypotenuse c is 30 mm and the side b is 27.5 mm, then the side a can be calculated as:

Pythagorean Theorem

c squared = 900.0 and b squared = 756.25

... so, the side a can now be calculated as well (symbol represents the square root):

$$a = \sqrt{30^2 - 27.5^2} \quad a = \sqrt{900 - 756.25} = 11.98958 \quad a = 11.990$$

Solving Right Triangles

Regardless of the actual method used, solving right triangles is a major part of manual programming. Calculations use the *sin*, *cos* and *tan* trigonometric functions, as well as the *Pythagorean Theorem*. As with any calculations, start with the known data. In trigonometry, any triangle can be solved, providing one of the two data sources is known:

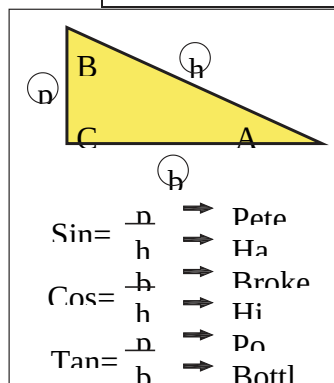
Two sides of a right triangle

One side and one angle of a right triangle

The 90 angle is never used in calculations. *Figure 53-22* shows trigonometric relations. If more than one solution is available, use both methods to double check the result.

Trigonometry Relationships
SOH

Opposite Sine = _____ Hypotenuse
CAH Adjacent Cosine = _____ Hypotenuse
TOA Opposite Tangent = _____ Adjacent



	a /	$c = \sin A = \cos B$
	b /	$c = \cos A = \sin B$
	a /	$b = \tan A = \cot B$
	b /	$a = \cot A = \tan B$
$a = c \times \sin A$	$b = c \times \cos A$	$c = a / \sin A$
$a = c \times \cos B$	$b = c \times \sin B$	$c = a / \cos B$
$a = b \times \tan A$	$b = a \times \tan B$	$c = b / \sin B$
$a = b / \tan B$	$b = a / \tan A$	$c = b / \cos A$
$a = \sqrt{c^2 - b^2}$	$b = \sqrt{c^2 - a^2}$	$c = \sqrt{a^2 + b^2}$
$A = 90^\circ - B$	$B = 90^\circ - A$	$C = 90^\circ$

Figure 53-22

Trigonometric functions - formulas for solving right angle triangles

J. ADVANCED CALCULATIONS

The last two illustrations show formulas to calculate the chord C or the tangent T of a circle. Trigonometric formulas can be used as well in this case, but well developed formulas can make the same calculations faster. With only one exception, there are two solutions, dependent on the available data. The formulas can also calculate the radius R , angle A and the deviation d . Calculations relative to the **chord** of a circle are shown in Figure 53-23. Calculations relative to the **tangent** of a circle are shown in Figure 53-24.

K. CONCLUSION

In this math for CNC oriented chapter, only the most important and commonly used mathematical subjects have been presented. Many more solutions and shortcuts are used by programmers and operators every day, showing their ingenuity in solving math problems. The author will appreciate any formula, shortcut or a practical solution to any programming problem - it will be seriously considered for the next edition of this digital book.

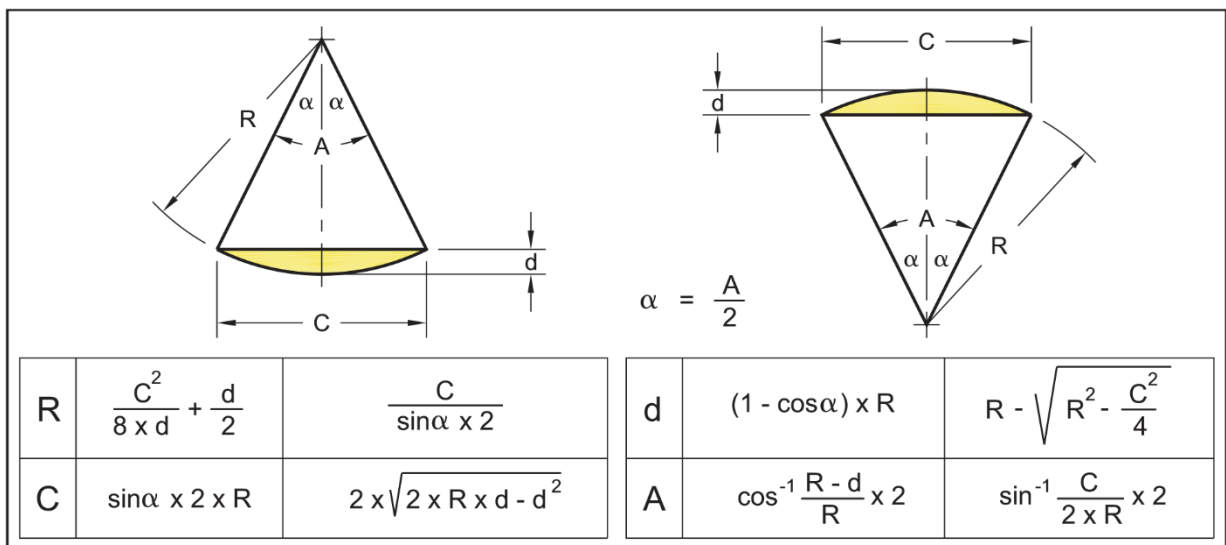


Figure 53-23

CHORD of a circle - calculations of chord, radius and deviation

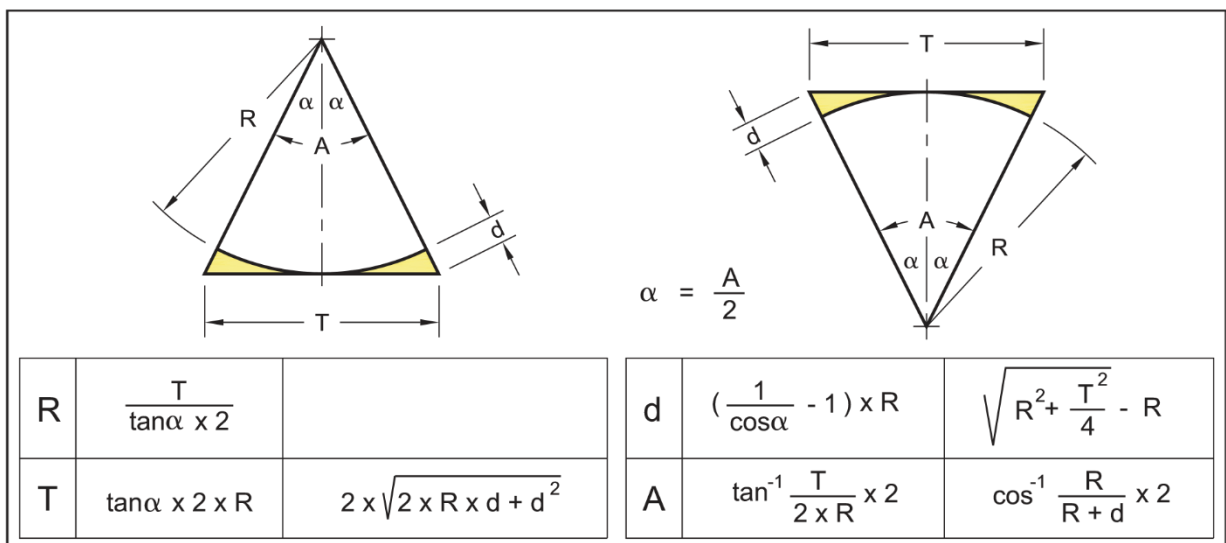


Figure 53-24

TANGENT of a circle - calculations of tangent, radius, angle and deviations

40. CNC AND CAD/CAM

Up to this point, all topics related to *manual* programming of CNC machines - all fifty-three chapters. In the last chapter, we look briefly at an area where manual programming is replaced by a computer, a suitable software and some *additional* skills. Note the word *additional*. Studying the digital book has certainly not been a waste of time. On the contrary - the digital book covers subjects that every CNC programmer should know, *regardless of the programming method used*. Programming with a computer is always desirable but to know the basic skills is the most important prerequisite. All basic skills are in understanding the manual process. All subjects and methods learned do not have to be applied by a pencil and paper. They could be applied by a CAD/CAM - or just CAM - programming. A simple statement may summarize it all:

Top class programming using CAM software requires solid knowledge of manual programming methods

PROGRAMMING MANUALLY?

In the area of CNC programming application techniques, computers at all levels, from a personal computer to workstations are capable to produce most CNC machine programs in a time much shorter than any manual programming method. So, why is the high importance of manual programming methods so emphasized? Is the manual programming still alive, and if so, how healthy is it?

There are at least two important reasons why manual programming for CNC machines it is not dead yet and will not disappear anytime soon.

The first reason is that in manual programming, the programmer is able to do what computers cannot - and never will be - *programmers can think*. Manual programming teaches the invaluable lessons of discipline - a very important quality of a professional CNC programmer. Discipline means to concentrate, to constantly evaluate, to make decisions - to think all the time. In manual programming, there is a total, absolute and unequivocal control over the final product - the part program. Only a programmer can evaluate a given situation, analyze the problem and adapt to unforeseen circumstances. Only a programmer can feel that something may not be right. Only people use instruments known as thinking process, intelligence, instinct, gut feel, common sense and experience. Those are instruments inherent to humans, not computers. CNC programming is like the work of an artist - it can never be fully automated.

A. CAM SOFTWARE

Current CNC software, commonly known as *CAM software*, has many features that translate into a CNC program, corresponding to individual ideas of how the part program should be written. It can produce a program closely matching a particular direction of thinking, closely matching a particular programming style. But *closely* does not always mean close enough. Here comes the second reason.

The second reason is that when programming manually, the programmer *understands* the programming process and the resulting output. A program generated by a computer has to be in the format compatible with the CNC machine and its control system. If all goes well, there is no need to look at the program at all - it's there, in the files, ready to be loaded into the CNC machine. On the other hand, what if there is a problem - what then? Going back to the computer and reprogram a part may solve the problem on hand. The question is at what price. Ability to read the CNC program code, to really understand it, also means the ability to change it. Spending a valuable computer time just to add a forgotten coolant function seems excessive. Would it not be better just to edit the program by adding M08 function in the right place? Although the example is oversimplified, it also shows that real *understanding* of the programming process is very important. The best way to understand this process is to bypass the computer and get the same results. That can be achieved with manual programming.

It would be unfair to compare or promote manual programming against computer programming and vice versa. What is necessary to promote is the knowledge and understanding of manual programming principles. Without such knowledge, one cannot become a good CNC programmer.

Most of the CNC programming can be done quite well on personal computers. The existing technology is progressing very rapidly and many 2D and 3D programming applications are available for a fraction of the cost when compared to just a few years ago. This trend will continue well into the future.

B. DESKTOP COMPUTER PROGRAMMING

Complete computer system - that means the hardware, software and peripherals - suitable for CNC programming is changing at such a rapid pace that any in-depth discussion of the hardware would be obsolete in a matter of weeks. Almost the same speed of obsolescence applies to software as well. New features, new capabilities, new tools

are constantly arriving on the market and are offered to users in both areas. Because hardware and software has to be considered together, the first question is what to select first - the hardware or the software?

Such a decision must be based on the required application. What will the computer be used for? What kind of work needs to be computerized, automated? What results are expected? These are the *primary* considerations - *not* the kind of monitor or printer or the capacity of hard disk. They are also very important - but only *after* establishing the application needs.

Certain programming applications are typical to all machine shops. Others are unique to a particular type of manufacturing and the kind of work or the product manufactured. The following short list itemizes the major groups that a typical computer-based CNC programming system should have:

- Tool path geometry creation environment

- Tool path generation

- Complete programming environment

- Post processing

- Training and technical support

It is important to understand *why* these features are important. Before investing into a technology that is all or partially new to the user, it helps to know *what* tools the software offers and *how* they can be used in everyday work.

C. TOOLPATH GEOMETRY DEVELOPMENT

Most CNC programming systems require a toolpath geometry creation before the actual path of a cutting tool can be generated. The key words here are *toolpath geometry*. A common misconception among programmers is that they have to re-create everything in the original drawing. That is a wrong approach.

When it comes to toolpath geometry, two scenarios must be faced. One will be work from a paper drawing, the other from a CAD drawing stored in the computer. Although there are differences in approach, the fact remains that either a new geometry is created, or an existing geometry is modified.

Modern CAM systems allow drawing in a CAD like manner - using editing features such as trim, fillet, break, copy, move, rotate, offset, mirror, scale, and so on.

Typically, programmer will define what is normally not on the drawing, at least not on a two dimensional representation of the part. Adding depth, separating entities by color, by levels

(layers), adding clearances or a special lead-in and lead-out tool motion, and so on.

D. TOOL PATH GENERATION

The key requirement of a CNC software is to produce a program of an accurate toolpath for a specific CNC machine. Toolpath creation, with all its calculations, is the most time consuming task in manual programming. It makes sense to make it the most important item to consider when planning to automate CNC programming process. Only high level CNC software supports a large variety of toolpaths. For example, helical milling or a full 3D machining are not always standard features in the software.

One mistake in software selection is to consider only the *existing* CNC machines and existing machining methods and practices. This rather narrowly focused approach is not always successful. Consider future plans in both strategies and capital investment. What about the product? How will the product change in five years? Knowing the philosophy and focus of the company, its policies and management strategies and yes - even its politics - will help to make a more accurate estimate of future needs.

Computer technology has grown a lot, yet it is so new that it is in the state of constant development. Nobody can predict with absolute accuracy what the future will offer in terms of CNC machining and CNC programming. If the current and the future needs are well established *before* purchasing a programming system, there is a good chance to beat obsolescence for a long time. CNC software developers offer periodical updates to their product, with more features added as computing power increases. The updates (new versions of the software), usually reflect developments of the technology, both on the hardware and software sides. It does not mean purchasing every new update offered, but it is important to select a CNC software developed by a solid and well established company that has the best chance to be still in existence when the need to update a system comes up. Computer industry is very active, and mergers, acquisitions and takeovers are as common as bankruptcies and failures.

E. COMPLETE ENVIRONMENT

A typical high quality CNC programming software allows all programming and relating tasks to be done from a structured menu, using a mouse or similar pointing device. The important thing is that once the software is loaded, it can complete all tasks without returning to the operating system level. Some programming systems are based on modules and files that are not accessible from a menu, or they do not cover all the steps in programming.

The following list is meant only as a very brief guide to some of the main features that apply to CNC programming on personal computers. These are the expected features from any CAM software:

- Multi machine support (machining centers, lathes, EDM)
- Associative operations for flexible editing
- Job setup and material blank definition
- Tooling list and job comments (setup sheets)
- Connection between computers
(communications feature)
- Program text editor (with CNC oriented features)
- Printing capabilities (text and graphics)
- Pen plotting (plotters)
- Interface with CAD software (DXF, IGES, STEP, STL, ...)

Support for solid modeling (third party)

Software specifications and features

(Including customizable post processing)

Support for generally available hardware

Utilities and special features, open architecture

Each described item will point out its significance. Although all items are useful in a programming system, it does not mean that all items are always necessary. Some features require an additional hardware equipment, such as a printer, plotter, cabling, small peripherals, etc.

Multi Machine Support

When it comes to support of different machine types and models, CNC software can be divided into two groups:

Dedicated software Integrated software

Dedicated software supports only one kind of machines. For example, a software that is designed specifically to produce programs for CNC fabrication equipment, cannot be used for lathes, machining centers or EDM.

Dedicated software is often developed for a rather narrow and very specialized field of applications or when it applies to a particular machine only. CNC punching, forming and press brake equipment are good examples of such software.

Integrated software allows the programmer selection of several types of machine tools. Such a selection usually offers milling, turning and wire EDM. It is also common to use the software for machines such as burners, routers, laser cutters, waterjets, and profilers. For metal cutting, this is the preferred type of software.

Another reason also speaks clearly in favor of integrated software, and that is its *interface*. It is much easier to get used to one display for a lathe work and have almost the same display for milling work or EDM work. Software menus look the same, the navigational operations share common menu items, customization of the software (including post processors) is much simplified.

Associative Operations

When a toolpath is developed, it is attached to the previously defined toolpath geometry. For many reasons, it is not unusual to change the toolpath geometry later. The traditional method has been (and for many software vendors still is) to recreate the geometry, then recreate the toolpath.

Associative operation avoids creation of a new toolpath, it updates it automatically. It is fast and accurate. It works the other way as well - many tooling parameters can be changed quickly, on demand.

Job Setup

Job setup is a feature that describes the material blank of the part - its shape, dimensions, zero origin, and many other related items. Tools and related speeds and feeds can be often selected from the job setup, as well as various program parameters. Libraries that store common data for tools, materials and operations are also powerful software features.

Tooling List and Job Comments

CNC programming is a process covering several steps. Whether programming manually or with a computer, the selection of cutting tools is a manual task. Once selected, each tool is assigned its identifications, speed and feed values. Several tools can be grouped into a tooling library file and stored. Then, the order of their usage within the program is selected. Some parts require more than one machining operation. Complex setups require special instructions to the machine operator

(setup sheet), describing the programmer's intents. All these programming decisions must be recorded, and the documentation sent out to the machine shop. It is only reasonable to expect that any CNC programming software will support a tooling list, perhaps in a form of a tool library file and the process list. Material library file is also very useful, as it can store surface speeds for many materials and the programming software will calculate the exact spindle speed and feed rate, based on the tool selected. This is a good example of interaction between the tool library and material library.

Connection Between Computers

A programming system should also include a connection (communications option) between the personal computer and the CNC machine. This feature allows the program data exchange via a cable. Programs can be sent from the computer to the memory of the CNC machine and back.

An important point is that not all CNC machines have the port (outlet) and the capability to take advantage of direct connection. Even if all machines in the shop have this capability, it requires additional hardware and organizational discipline to make all elements work in harmony. The existence of a direct connection in a programming software is a must, even if it is not used immediately after the purchase.

Program Text Editor

A CNC program generated by the software should be 100% complete and ready for use by the machine. The implication is that such a program is so perfect that it needs no further editing. This is the ideal way, the way it should happen. If a change in the program is needed, it should be done *within* the design of the part shape and that means through the CNC software - *not outside of it*. The reason is that any manual change to the generated program does not correspond to the program data as generated by the computer. In the environment where the data is shared by many users, such a practice will cause a lot of problems.

That brings up a question - why does a CNC software have a built-in text editor? There are two reasons. One, the editor can be used for creating or modifying various text files such as setup sheets, tooling sheets, operation data, post processor templates, configuration files, special instructions, procedures, etc. These files can be updated and otherwise modified as required, *without* a damage to the program database. The second reason is that in some *special* circumstances, a CNC program can be edited outside of the computer, providing such change does not modify significant data. For example, to add a missing *coolant* function M08 to the part program is much faster done in the text editor, than repeating the program generating process with the computer. Purists are right, it is not the right way of using a text editor, but at least the significant data (tool locations) are not tampered with, and the database is otherwise completely accurate.

Many programmers use various external text editors or even word processors in text mode. These types of editors are not oriented towards the CNC programming, since they lack some features typical to the CNC program development. Only a CNC oriented text editors can handle automatic block number sequencing, removing the block numbers, adding cosmetic spaces in the program and other functions. The editor should be accessible from the main menu or from within the software.

Printing Capabilities

Any text saved into a file, CNC programs included, can be printed using a standard printer. Paper copy is often necessary as a reference for the CNC operator, for stored documentation, or just for convenience. Printer does not need to be top of the line, just one with a standard paper width. Some programming software supports an option that is known as a printer plot or a hard copy. Hard copy is a graphic image of the screen transferred to the printer. The image quality is usually more than adequate. This hard copy is an excellent aid during program development stage. Better quality printer provides better quality print plot. The printer support is provided by the Windows environment, as most PC based CAM software is developed for the Windows operating

system.

Pen Plotting

Pen plot will usually produce image quality superior to the printer plot but for a CAM programming it is an unnecessary luxury. The only time when a pen plotter can be beneficial is for plotting to paper size that is not supported by standard printers. Other reasons will be the need for a color output, a special requirement by customers, or special documentation development. Before the graphics software appeared on the market, plotters were widely used to verify the tool path. Now, the toolpath is verified directly on the computer display screen, during interactive programming process, including different views and zooms. Printing can utilize monochrome or color laser printers. Most plotters are HPGL compatible. HPGL is an acronym for *Hewlett-Packard Graphics Language* and is currently the most supported plot file exchange format.

CAD Software Access

If an engineering drawing is generated by CAD software, all drawing information is stored in a computer database that can be accessed by other software, through a file format translation utility (more on the subject later). Once the CNC software accepted and processed the database from a CAD system, the CNC programmer can concentrate on generation of the toolpath itself, rather than defining toolpath geometry from scratch. Some modifications are usually necessary, so expect them. The most significant advantage of a quality CAD/CAM system is the avoidance of duplication. Without CAD system, CNC programmer has a lot of extra work to do, much of it is duplicated.

A high quality CNC software also allows the existing program file to be translated the other way, to a file that a CAD system can accept. This option is called *reversed processing* and can be a benefit to companies that want to translate existing programs generated manually to an electronic form. Usually, some additional work is required in these cases.

High level CNC software is a *standalone* type. Standalone software means that it does not need an access to a CAD system - toolpath geometry and toolpath itself can be developed from within the CAM software, independently of other software.

Support for Solids

Solid modeling for 3D applications had been for a long time the domain of large computer systems. With the advance of powerful microcomputers, solid modeling is now an optional part of high level CNC software.

With solid models, the machining process of complex surfaces is much more streamlined. In addition, solid models offer the benefits of supplying engineering data, easier manipulation of objects, and many other features.

Software Specifications

Another benefit of a high level CNC software is that it comes well supplied with a variety of useful features. What makes each system unique, is usually the method of how the programming process is executed. In the early years of development, programming was done by using special programming languages, such as APT™ or Compact II™. Some languages are still available but heavily on the decline. Modern interactive graphics programming has virtually eliminated the need for languages in just about all manufacturing fields. The more popular kind of programming is based on *interactive graphics*. Programmer defines geometry, typically as toolpath geometry, followed by the toolpath itself. Any error in the process is immediately displayed on the graphic screen and can be corrected before too much other work is done.

Hardware Specifications

Specification of the software will determine the hardware selection. Hardware is a common term for the computer, monitor, keyboard, printer, modem, plotter, mouse, scanner, disk drive, storage media, CD/DVD writer, and many others. Hardware referred to in this chapter is based on

the *Windows*TM operating systems. Modern operating systems are based on a *graphical user interface (GUI)*. Some software can run under a different operating system, for example Unix (used mainly by large workstations) or different Windows versions. It is always to the user's advantage that the latest version of the operating system and CAM software is installed on the computer.

When thinking of purchasing a computer hardware, consider carefully at least three major criteria:

Performance	... computer speed
Data storage	... type and size
Input / Output	... ports
Computer Speed	

Performance of the computer system is typically measured by the relative speed of the main processor. The higher the number, the faster the computer can process data. To make the comparison easier, the original IBM PC, model year 1983, had a 4.77MHz processor speed. Later model AT had 6MHz processor speed, improved further to 8 and 10MHz. Later, computers used the so called 386 microchip (generally Intel 80386 or 80486) and reached 25MHz, 33MHz and more. Pentium processors followed, and the process is ongoing. For serious CAD/CAM work, the latest fully featured processors should be used. Newest processors offer much higher processing speed, measured in GHz, and the more processing speed is available, the better performance of CNC programming system. Fast graphics card is also of the highest importance.

RAM and Data Storage

Data is stored in the computer in two forms - memory storage and disk storage (file). When an application such as CNC programming is started, CAM software is loaded into the computer memory. The more powerful the application software, the more memory it requires. This memory is known as *Random Access Memory*, usually called *RAM*. Every software specification identifies the minimum available RAM required. RAM of today high level computers around two to four gigabyte range is not uncommon. Any extra memory will speed up processing quite significantly. Data in the RAM is volatile, which means it is lost when the application is ended, or the computer power is interrupted. To save important data from RAM into disk files, a hard disk or similar media can be used. For a microcomputer CAD/CAM work, the absolute minimum requirement is high density removable drive and one large size hard drive. Floppy drives of any kind are not suitable.

The hard drive should have a fast access time and a high storage capacity. Another option is a tape drive, CD-R and CD-RW disks or recordable DVD disks for backup.

Input and Output

Input and Output (I/O) computer features, cover hardware items such as monitor, graphic card, keyboard, digitizer, scanner, printer and plotter. Monitor suitable for CAD/CAM work should be a large size color monitor providing very high resolution. The monitor and graphic card do relate to each other. The card must be able to generate the image, the monitor must be able to display the image. Speed of the video output is also very important.

A keyboard is a standard feature of a computer and serves as a basic input device. Mouse (or a digitizer on larger systems) are also input devices, but much faster than keyboard input. In CAD/CAM, where much work is done in graphic mode under a menu system, an item from the menu is user selected. In most cases it can be selected with a pointing device. User points at the menu item desired, presses a button on the device and the menu item is executed. Pointing device most suitable for CAM work in the Windows environment is a mouse with several buttons and a scroll wheel.

Both the printer and plotter are theoretically optional, but generally worth some consideration.

For CNC work alone, a printer is more important than a pen plotter. If the setup is a true CAD/CAM, both peripheral devices may be needed.

All peripherals are interfaced with the computer using specially configured cables connected to the *Input/Output (I/O)* outlets called *ports*. Modem is normally not required for CNC programming, except for data exchange with a remote computer or Internet access. Laser or ink jet printers generally use a parallel interface known as the *Centronics* standard, but many other devices use a serial interface. There are also other popular I/O options, such as the USB (Universal Serial Bus) or Firewire connection.

Typical Hardware / Software Requirements

Currently, the most popular hardware for CNC programming is the Windows based computer system. It is not possible to make a simple 'shopping list' for all hardware requirements that every CNC machine shop can use. Here are some rules applicable to any system and are not subject to becoming outdated very quickly. A typical list of minimum hardware requirements and options may be compiled:

- Hardware compatibility with IBM (Windows based)

- Apple computers have very limited CAD/CAM applications

- The latest version of the Windows operating system

- (Must be supported by the CAM software)

- High central processor speed - higher = better

- (Measured in Gigahertz units - GHz)

- Fast memory cache

- Requirement of a numeric (math) processor, which is normally part of most main processors

- Random Access Memory (RAM) - as much as possible

- Enough of hard disk space for program and data storage

- (Measured in gigabytes or higher - with a fast access time)

- Backup system for data protection

- (Tape cartridge, removable drive, CD, DVD, ...)

- High resolution graphics adapter (graphics card)

- (Should have a rapid refreshing for the video output)

- Large high resolution color monitor - non-interlaced (measured in pixels - the more pixels per screen size, the finer the display, and the smaller the pixel size, the better the display)

- Pointing device - normally a mouse - is a current standard

- Pen plotter is required only in special circumstances (not needed for CNC work) - B size maximum is usually enough, if needed

- Working real time calendar clock (stamps all created files with the current date and time - standard feature)

- A good quality printer with a parallel or USB port

- (For hard copy documentation)

- CD or DVD drive & various multimedia features (sound card necessary)

- Access to additional global information

- (Internet, E-mail, user groups, newsgroups, ...)

Two or more serial and USB ports

Text editor - usually part of the software (or optional)

It is smart to keep abreast of the micro computer technology. It develops rapidly and even a few weeks may change some fundamental approaches and decisions. Following the development of computer technology creates awareness of the latest improvements, therefore a more educated user and/or buyer.

Utilities and Special Features

Even the most updated version of the operating system is never as powerful and flexible as many users would like it to be. For that reason, many software developers came up with literally thousands of programs and utilities that supplement the readily available features. Many of these utilities are available as shareware or freeware from the Internet and other sources. Access to the Internet and the World Wide Web provides a great source of CNC and machine shop related topics and general information. These utilities are not necessary to use a CAM software, but they are a great time saver for many tasks associated with using a computer.

F. POST PROCESSORS

CNC software must be able to output a program in a format unique to each control unit. The most important part of toolpath generation is data integrity. Computer generated program must be accurate and ready for the CNC machine. That means the completed program should require no editing, no optimization, no merging with other programs or similar manual activities. Such a goal can be achieved only by a well-developed programming style - and a properly configured *postprocessor* for each different CNC machine.

A top quality post processor is probably the most important customized feature of a CNC software. When entering data into the software, settings describing the part shape, cutting values, spindle speeds, and many other data are stored for further processing. The software analyzes this data, sorts it and creates a database. This database represents the part geometry, toolpath geometry and other functions. CNC system cannot *understand* the data, regardless of its accuracy. To complicate things even more, every CNC system is different. Some program codes are unique to a single machine, some are quite common to many machines. The purpose of a post processor is to process the generic data and convert them to the machine code for individual control systems.

Customizing Post Processor

Typically, a supplied post processor is more or less generic and has to be customized, at least to some extent. To develop a post processor in-house, usually means to customize the generic post processor supplied with the CAM software. Typical process depends on the type of post processor and its format. Small changes may take minutes, large changes days. Post processors can be very expensive.

CNC programmer must know the machine and control features extremely well. A deep and thorough knowledge of manual programming methods is a must - how else can a useful programming format be developed? Also important is the knowledge of machining methods. Finally, knowing any high level language can make the post processor development much more efficient and powerful.

G. IMPORTANT FEATURES

There are several important features to look into when investing into a CNC programming software. They do have an impact on the final functionality of the program, at machine level. All these features are important and should be considered carefully.

Input from User

One of the important features of a CAM programming software is its ability to handle input

from the user. This input can be a special sequence of commands that cannot be handled by post processor at all or would require too much effort. These commands are usually small in size and can be called and used in the graphics mode whenever required. Examples of such applications are a bar feeder sequence on a lathe or a pallet changing routine on a horizontal machining center. If the software supports some type of variable type of user commands, it adds an extra flexibility and power to the system.

Machining Cycles

Another very important feature of a CAM software is its ability to generate a variety of fixed and repetitive cycles, that modern controls support. These cycles make a manual programming simpler and faster. Modern CNC systems take advantage of such cycles are available with a limited memory capacity. For that reason, support for such cycles is very important in CNC software, as it provides easy editing at the machine.

Also important is the representation of the toolpath for lathe tools. Many cutting tools for CNC lathes have a back angle. A high quality software should also evaluate the tool back angle in its calculations and in the display.

CAD Interface

A standalone CNC programming system does not need a CAD software for geometry definitions. It can create its own. Yet, in any CAD/CAM system is important to have the *option* of importing part geometry *from* a CAD system. Even if a company does not need CAD, it should be prepared to accept its files, perhaps from customers or company branch offices. Support for acceptance of native files is becoming the new standard.

Needless to say, if programming software cannot accept native drawing files, another solution must exist. Files from different vendors are proprietary and their structure is not a matter of public access. There is another way - *use a different file format*.

File Exchange Formats

The need to exchange design files between different software systems has always been a prime requirement. There are many competing formats of a neutral file format. The oldest of them is called *IGES (Initial Graphics Exchange Specification)*, originally developed to transfer complex design files from one software to another. Another format that is also used, is the *DXF* format by Autodesk™.

The *DXF (Drawing eXchange Format or Data eXchange Format)* is considered by many to be *the* standard of drawing file exchange between microcomputers. It has been developed by Autodesk™, Inc., developers of the popular AutoCAD™, the most widely used PC based CAD in the world. DXF format is suitable only for common geometric elements, such as points, lines, arcs and a few others.

CNC software should also support an interface between the neutral files generated by a CAD system. Depending on the nature of a particular programming application, DXF interface may be needed for simpler jobs, and IGES for more complex geometries. High quality CNC software offers at least these two formats, usually many more. Keep in mind that the format and structure of the translators, such as DXF or IGES, is not in the hands of the CNC software developer, therefore it is a subject to change.

H. SUPPORT AND MANAGEMENT

Hardware and software for CNC programming work can be costly. It can represent a significant investment of money *and* people and can become a total failure if it is not used properly. A failure is not the actual loss of the hardware and software cost. The real and heaviest loss is in the increased productivity, speed and quality that was expected but never materialized. Loss is also in the

confidence the company employees put into the technology. These losses can be high. To prevent such prospects, keep three key elements in mind when planning a CNC programming system:

High quality training program for long term skills

System management philosophy and strategies

Technical support for hardware and software

No single item in the list is any more important than the others - they are *all equally* important.

Training

Training should be planned, thorough, and professional. Many successful programs apply three levels of training. Some companies do not place enough emphasis on training, despite many studies and examples proving that good quality training does work. The lack of time and perceived high costs are often used as excuses. Training is a necessary investment for any company that wants to be competitive.

Training Level 1

The *first level* of training should be aimed at the person with *none* or *very little* computer experience. It should introduce CNC software to the programmer who programs manually. It should be an overall training, mainly general in nature, with the main emphasis on system features and capabilities - as they relate to the company where the software is installed. Typical general approach should be balanced by explaining philosophy behind software design, and structure of menus and commands. It is very important to show the student what the software can do in skilled hands. The first level should be done when the software is purchased. The objective is to give the programmer enough tools to *play* with the software, to grow into it. A simple way to achieve this goal is to try out simple projects while still programming manually more important jobs.

Training Level 2

The *second level* is most beneficial two or three weeks *after* the first level completion. It is the most critical level of the three. It should include a *systematic approach* to all software features, with special emphasis on features relating to machining operations used locally. This training level eliminates manual programming and marks the beginning of a new era. Supervisors should evaluate the complexity of the first few jobs to be programmed and select, if possible, the less difficult jobs to build a little confidence.

Training Level 3

The *third level* is usually done 2-3 months later. It covers problems, questions, difficulties and concerns, introduces tips, shortcuts, etc. The purpose of this level is to create a long term confidence. At this stage, the programmer has many questions. Professional instructor can answer all questions, weed out bad habits, and offer further guidance.

System Management

A reliable operation of all system elements is crucial to the success of CNC software. Use of any software requires good organization, it needs strategies, it needs focus, and it definitely needs a professional management. System management establishes standards and procedures for CNC and related operations. Concerns about people selection, data backup methods, confidentiality and security, work environment quality, etc., are not confined to a single discipline and should be important in the overall company culture.

Technical Support

Technical support is an important part of system management. A service contract or a support package can be usually negotiated with the vendor, covering installation, hardware, update policies, new developments, information flow, etc. An important part of technical support is the speed and reliability of handling *emergency* situations. If a hard disk fails - and a data backup *does* exist - what can be done? CNC shop is waiting for the critical job, while the programmer cannot

send program data to the machine, because an inexpensive hard disk failed. Support should cover both hardware and software. All support promised by the vendor should always be written down. Know *exactly* what the bill is for. *If something isn't in the contract, it usually isn't available.*