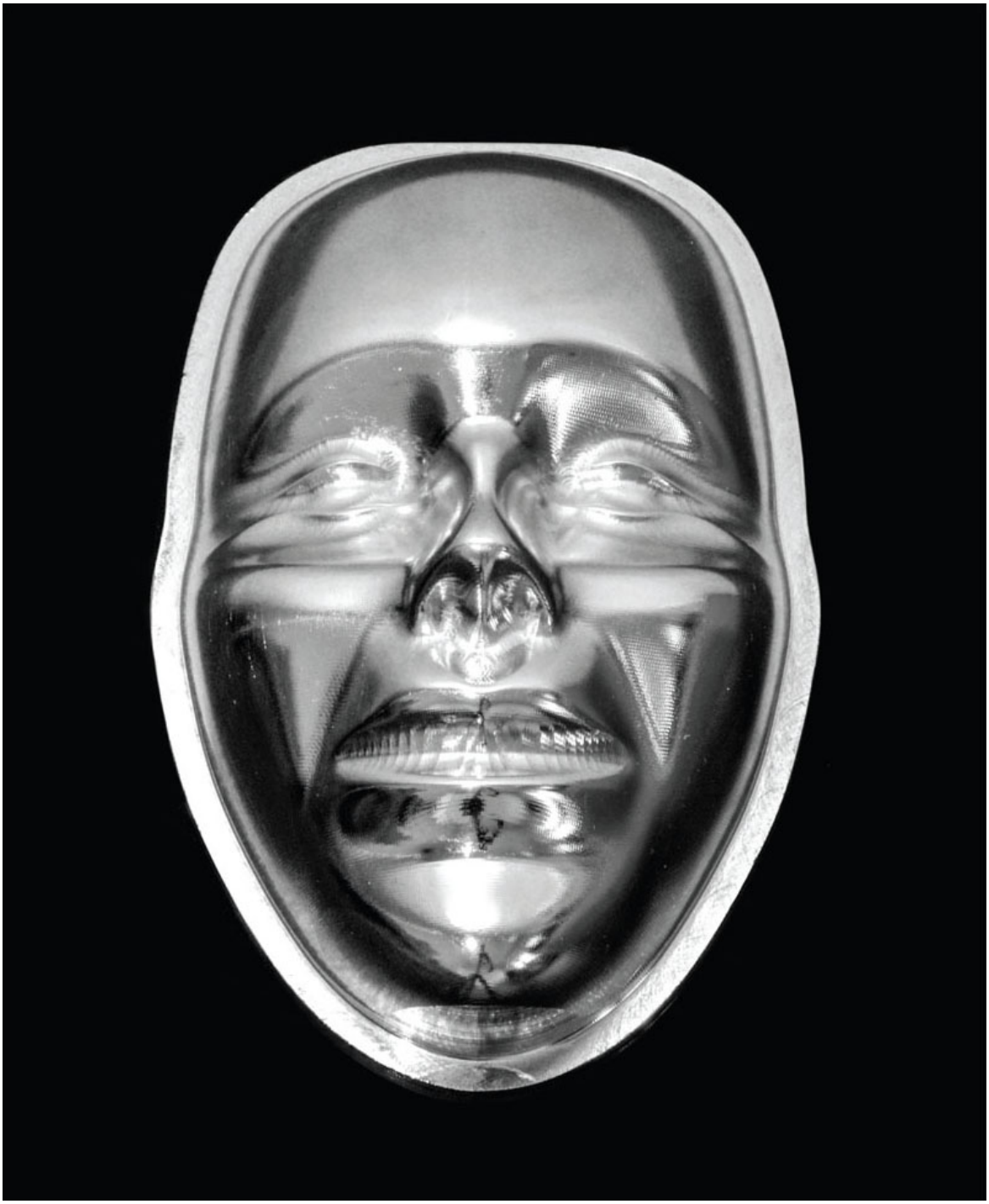
CROWOOD METALWORKING GUIDES

CNC MILLING IN THE WORKSHOP

DR MARCUS BOWMAN

CROWOOD METALWORKING GUIDES

CNC MILLING IN THE WORKSHOP



CROWOOD METALWORKING GUIDES

CNC MILLING IN THE WORKSHOP

DR MARCUS BOWMAN

THE CROWOOD PRESS

First published in 2013 by
The Crowood Press Ltd
Ramsbury, Marlborough
Wiltshire SN8 2HR

www.crowood.com

This e-book first published in 2013

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

ISBN 978 1 84797 630 7

Frontispiece: machined aluminium face by Tommi Salminen

Contents

Introduction

The aim of this book is to introduce a range of concepts and techniques for producing parts using a computer-controlled milling machine. This is a practical book containing techniques to put your own CNC machine tool to work.

The book explains the machines, the software and the methods for producing a range of parts varying from the simple to the complex and the functional to the artistic, and includes guidance on tooling, speeds, feeds and fixtures. Throughout the book there are a number of projects that you can use to try out the various techniques.

SOME ASSUMPTIONS

This is a practical book about using a computer-controlled machine tool to do useful work, and assumes you have a CNC milling machine set up to move under software control.

The book makes reference to Mach3, LinuxCNC, software from the Vectric range (including Cut2D and VCarve Pro) and software packages from other companies, but the techniques are applicable to most similar software.

A Word about the Hardware

This book assumes that you will be using a benchtop CNC mill designed for metalworking, or a larger knee mill such as a Bridgeport. A range of materials is used throughout the book, so it is not restricted to metalworking. The workpieces in the projects are relatively small and will fit on most benchtop CNC mills. This means

that they will fit on most gantry mills, of course. However, when it comes to chewing substantial lumps out of steel, most gantry mills are just not designed for that. If you use a softer material, they will work just fine.

A Word about the Software

This book does not provide detailed stepby-step instructions for any software package. There are detailed instructions for two of the most popular packages in the section on basic movement, just to make sure you can get your machine moving, but the rest of the book assumes you can read the software manuals for the packages you are using.

The book deals both with programming and with the use of software that will generate a program for you from a drawing. The program generators are easy to use and are essential for many jobs, but they can be used for very simple tasks too.

Programming by hand is another kettle of fish entirely, and simple jobs can often be programmed directly, instead of always having to use a program generator. Sometimes, too, the program generators cannot cope with the way a workpiece has been set up on the mill, so the technique there is to use the program generator to create the main parts of the program and to add a few lines of your own program instructions to link those elements.

So you, as the user, will need a knowledge of both programming and program generators. In the real world, one cannot wholly exist without the other. Besides, some knowledge of what is going on under the hood is useful.

Skip lightly across the deeper waters at any point; move on and continue reading. You may wish to come back later to any of the sections of the book as you gain experience and feel the need.

The machine control software packages Mach3 and LinuxCNC are used for the examples throughout the book. Both packages do the same job, but while Mach3 is a more graphically oriented system, with many accessible menus for setting up the links between

software and the machine tool, LinuxCNC provides a much more elegant programming environment.

Most CNC packages share a common core, and that includes many of the other CNC control programs that are available. By using examples from both Mach3 and LinuxCNC, this book should be applicable to most CNC control software currently available from commercial vendors as well as the packages being used by hobbyists who have built or assembled their own CNC systems.

It is quite possible to use both software packages with the same machine at different times, and reading about both software approaches might help you decide what is most appropriate for you.

Vectric software has been used in many examples in the book. Vectric has a range of software applicable to a lot of the techniques illustrated in this book. The Vectric packages are essentially program generators that can convert drawings or photographs into programs that can be used by a CNC control program to machine the end product.

A Word about the Approach

The approach throughout is focused on practical aspects of CNC machining. The book explains a range of techniques, from the simple to the ambitious, which can be used to machine various features on a workpiece. The simple techniques can be put to use straight away, while the more ambitious are there to encourage you to use your machine to the full. A manual mill can carry out a good range of machining tasks, and the point of owning a CNC mill is to push the boundaries of what can be machined on a workpiece. You might, for example, mill a simple rectangular lid for a box; but with a CNC mill, it is a very small step to engrave the top, give the edge a complex smooth and flowing curve, or texture one of the faces. These are all things that would be very difficult to do with a manual mill, but they really bring a piece of work to life.

There are suggested projects throughout the book that are designed to allow you to practise what you have read. Later projects

deliberately provoke thought. They are all there for you to enjoy.

A BIT OF HISTORY TO SET THE CONTEXT

The modern machine tool has its origins in the fifteenth century, but computer control of machine tools is a much more modern phenomenon, dating from the 1950s.

Hand tools have a longer history, and since early times, skilled artisans have used tools like files, chisels, hammers and scrapers to produce work that has sometimes been of astonishing accuracy and beauty. Accurate clocks and scientific instruments, for example, were initially produced using hand tools. Skill is an important factor, though, and two craftsmen working from the same set of drawings or instructions will inevitably produce work that differs in accuracy and finish, according to their individual levels of skill.

The rise of mass production in the early 1900s demanded that components produced by individual workers be sufficiently similar to allow interchangeability of parts so that a complete assembly, like a car or a firearm, could be made from parts produced by any worker.

This also meant that repairs could be made by replacing individual parts from a stock of standard parts. The demands of mass production led to the development of *standards for drawings, systems of measurement and tolerances on components, and repeatable accuracy in manufacture*. All of those are important for CNC machines.

A machine tool eliminates the variability associated with the human hand and eye. The milling machine and lathe use slideways and feedscrews to guide a tool accurately and repeatably, often at a more consistent rate, ensuring that *repeated movements produce repeatable results*. This is a basic principle behind CNC machines.

The development of the electric motor in the 1890s led to the self-contained machine tool, and more recent developments in electronics have led to the creation of servos and stepper motors that can be used to move feedscrews repeatably, predictably and with considerable accuracy. In mass production, a machine tool is

often used to produce many copies of the same part, so mechanical techniques were developed to control machine tools automatically using cams and levers. The cost of creating the cams, and the time taken to set up these mechanically automated machines, meant they were ideally suited to mass production, but were not economical for small quantities. Controlling a machine tool using a computer-based system means that the movements of the machine can be controlled by instructions in a computer program. Running the program guides the movement of the machine and it is only necessary to run the same program again to produce another set of identical movements. Changing the instructions changes the movements made by the machine and results in a different part being manufactured. This means that the same machine tool can be used much more flexibly, because different parts can be manufactured simply by changing the instructions in the computer program. An additional benefit is that complex shapes can easily be defined in software. This system is known as computer numerical control (CNC) and it brings significant benefits in *flexibility of manufacture*.

With developments in software, parts can be designed and drawn in two or three dimensions using computer-aided design (CAD) software. A computer-aided manufacture (CAM) software package can then read the CAD file and create the instructions for the program that controls a CNC machine tool, and the part can be produced by running that program. A CAD/CAM/CNC system provides flexibility and is capable of controlling a range of machine tools. In some instances, parts can be produced using CAD/CAM/CNC that could not be produced using conventional manual machining methods.

The availability of relatively inexpensive control systems based on personal computers, servos and stepper motors means that second-hand industrial CNC machine tools, smaller inexpensive purpose-made CNC machines and home-constructed CNC machine tools are within the reach of a wide range of users. Alongside developments in control systems, standardized software packages allow anyone to use these CNC machines to produce complex parts

with ease, in batches ranging from a single part to tens of thousands of similar parts with a considerable degree of accuracy.

The workshop has never been a more exciting place.

A medium-sized benchtop mill: the KX3 from Arc Euro Trade.

1 Fundamental Concepts

In this chapter you will learn about:

- the process of getting from a design to a finished workpiece;
- details of the software systems you might use;
- some of the mechanical and electronic systems used in CNC milling machines.

FROM DESIGN TO COMPLETED PROJECT

The traditional explanation of what happens in the workshop is that an idea is turned into a pencil sketch on the back of an old envelope, and a machinist uses that information to produce a finished workpiece while standing at the mill.

That romantic and rather unrealistic view of the process needs some refinement for a CNC machine. Fig. 1-1 shows the stages in the journey from design to completed workpiece, indicating the relative contribution of a human (on the left) and a computer (on the right). The original idea, for example, is an entirely human contribution, whereas creating the program may require both human and computer contributions.

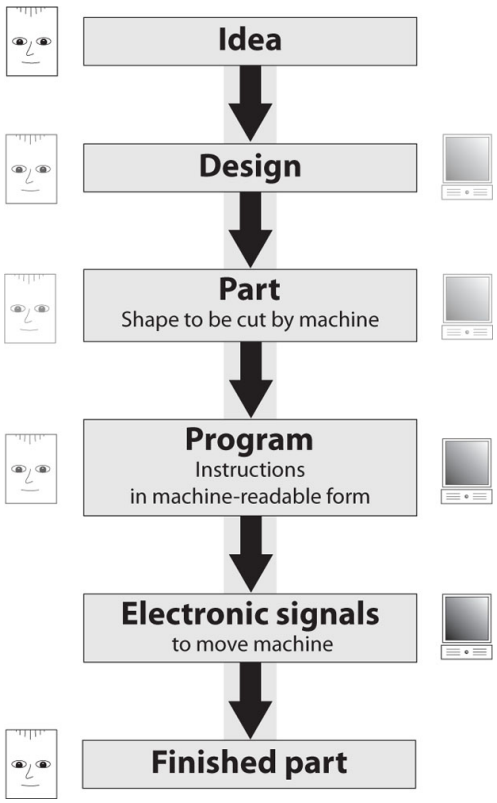


Fig. 1-1 The main software functions associated with a CNC system.

From a software point of view, there are three main stages in the process: create a design, turn the design into data and use the data to control a machine. These correspond to the conventional computer-aided design (CAD), computer-aided manufacture (CAM) and computer numerical control (CNC) stages shown in Fig. 1-2.

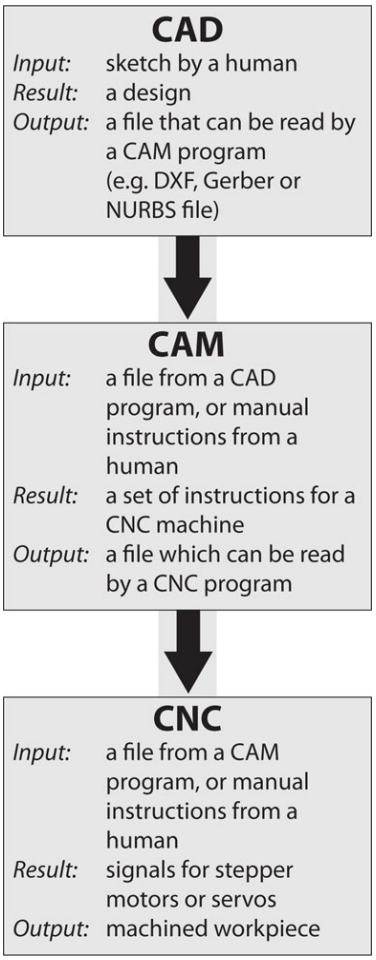


Fig. 1-2 Stages in the CAD/CAM/CNC process.

Creating a Design

Unlike the rough sketch on a small piece of paper, the design stage must result in a completely specified design in which all the geometric features are present, accurately sized and precisely positioned. Whether the design is produced on paper or on a computer monitor depends on the complexity of the design and the nature of the features.

Some designs consist of simple arrangements of easily defined shapes such as straight lines and circles, but anything other than the

simplest of shapes will benefit from the use of appropriate design software.

The design shown in Fig. 1-3 is a simple shape that could be fully specified on a small piece of paper and hand coded directly into the editor of a CNC package without much trouble.

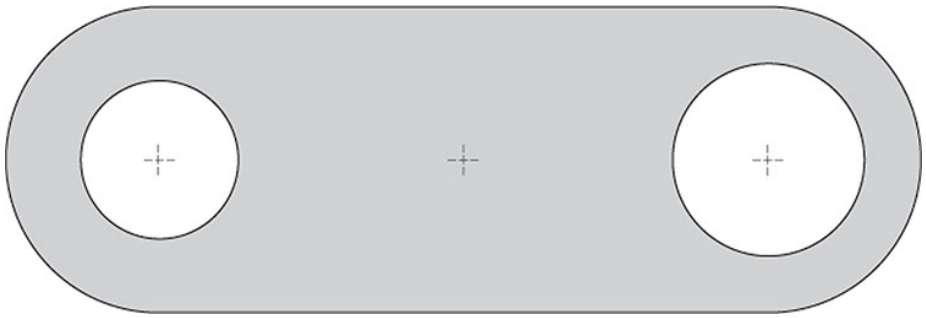


Fig. 1-3 A geometrically simple shape.

The design shown in Fig. 1-4 is an apparently simple design that can easily be sketched on paper, but cannot be fully specified without some significant mathematical calculations to identify the intersection points at A and B. Without those points, there is insufficient information to be able to create a CNC program.

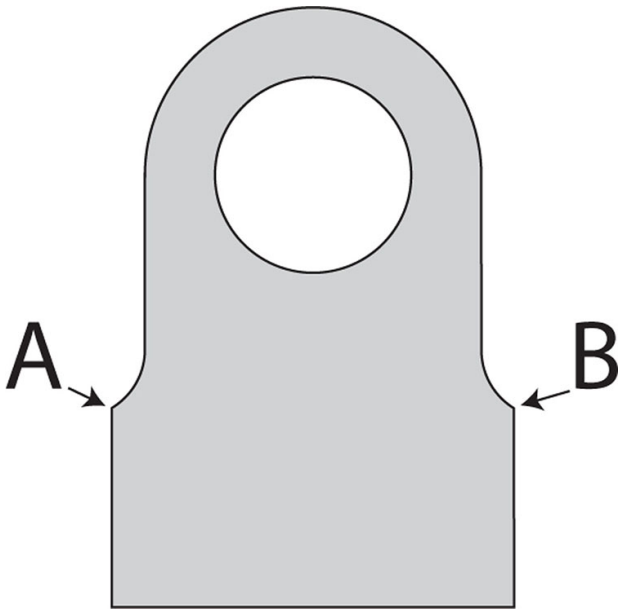


Fig. 1-4 A shape that appears simple, but includes a geometric complexity.

The design shown in Fig. 1-5 is of a visually and geometrically rich object (a set of clock plates) that might be sketched on paper, depending on the skill of the artist, but because of the complex curves, it would take a very considerable effort to fully specify the geometry and a mastery of advanced mathematics to calculate and prepare the data required for a CNC machine.

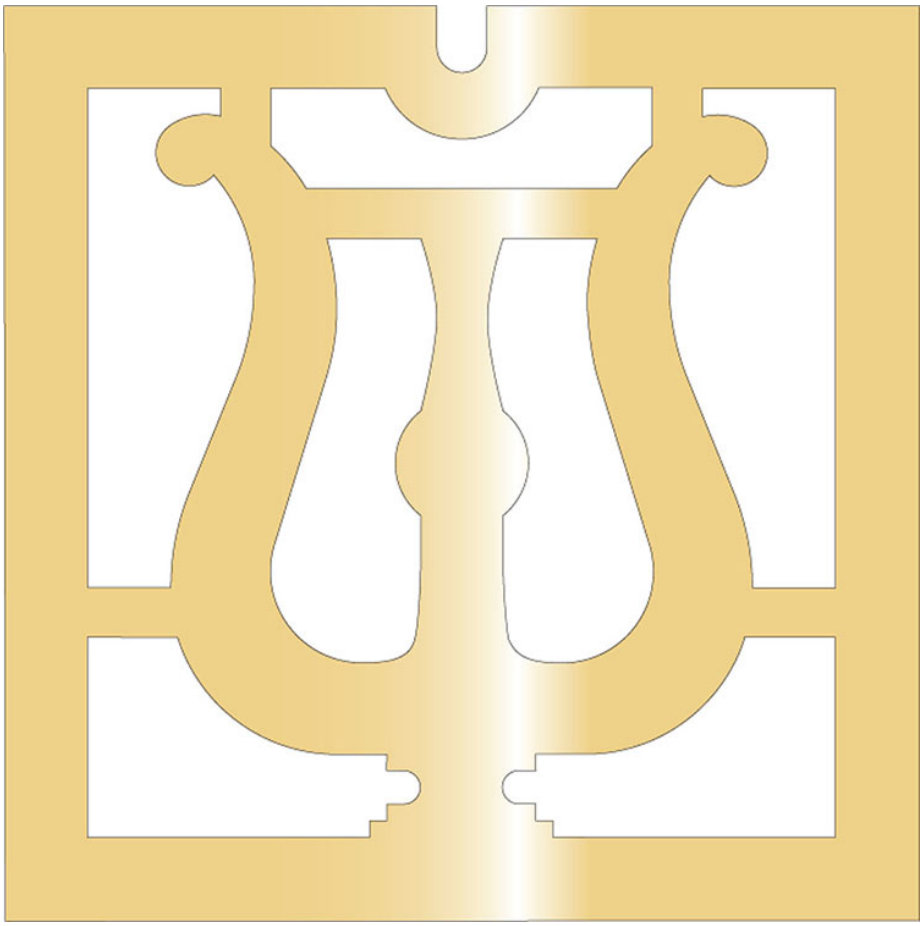


Fig. 1-5 A clock plate – a geometrically rich and complex shape.

In fact, the design process itself is severely compromised if it is restricted to designs that can be sketched on paper and hand coded

as CNC programs. On the other hand, consistent use of CAD software allows most designs to be created with relative ease and for those designs to incorporate features that might enhance the visual appeal of the design, in the knowledge that those additional features can be produced easily in the CAM and CNC stages of the process, usually for little additional human effort.

Figs 1-6 and 1-7 show two versions of a design for a support bracket for a toolpost grinder. The first, in Fig. 1-6, is based on a geometrically simple shape that can be specified on paper and hand coded. The second version, in Fig. 1-7, contains enhancements to the design to improve the usefulness of the object as well as its aesthetic appeal. This second version was designed in a CAD package, coded by CAM software, and machined using a CNC program. Fig. 1-8 shows the bracket in use. Once drawn in a CAD package, even geometrically complex objects can easily be produced by employing a CAM package to do the sophisticated mathematics required for the toolpaths and a CNC package to follow the instructions generated by the CAM package.

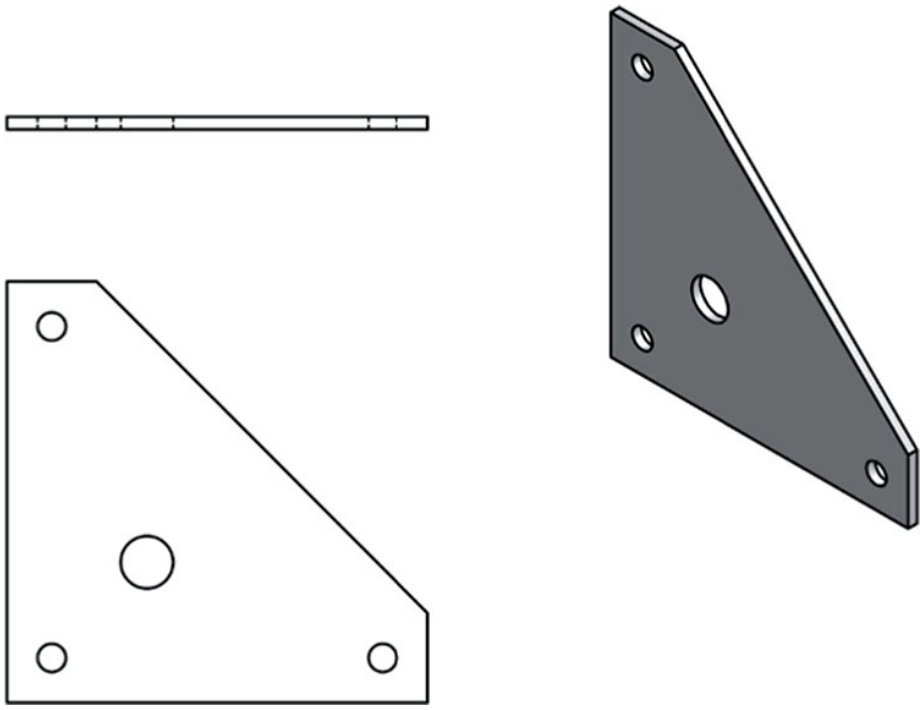


Fig. 1-6 A simple bracket.

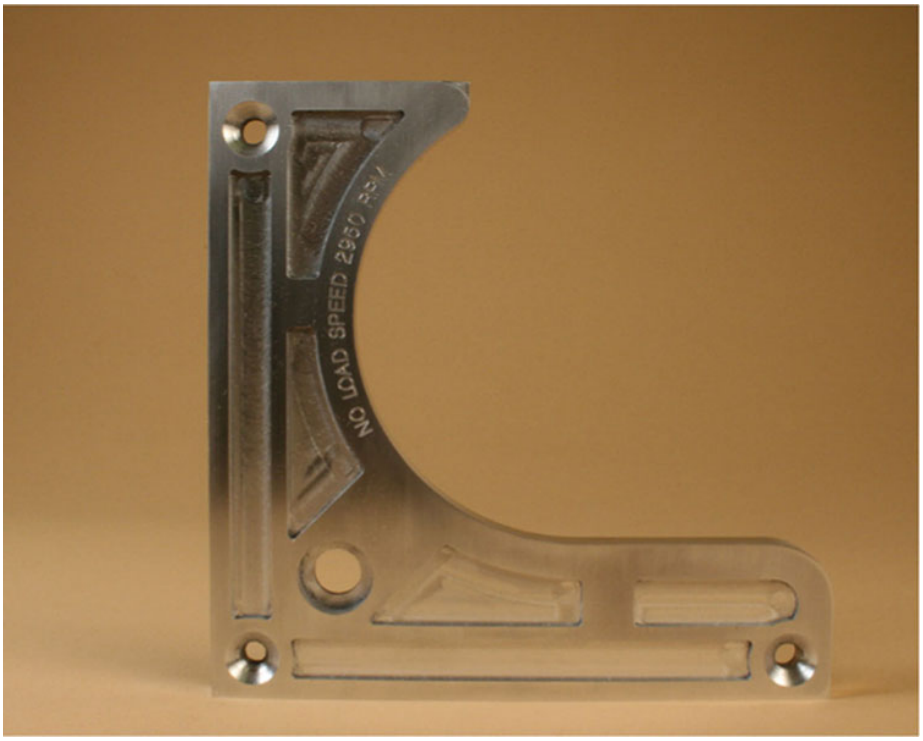


Fig. 1-7 An enhanced version of the bracket.

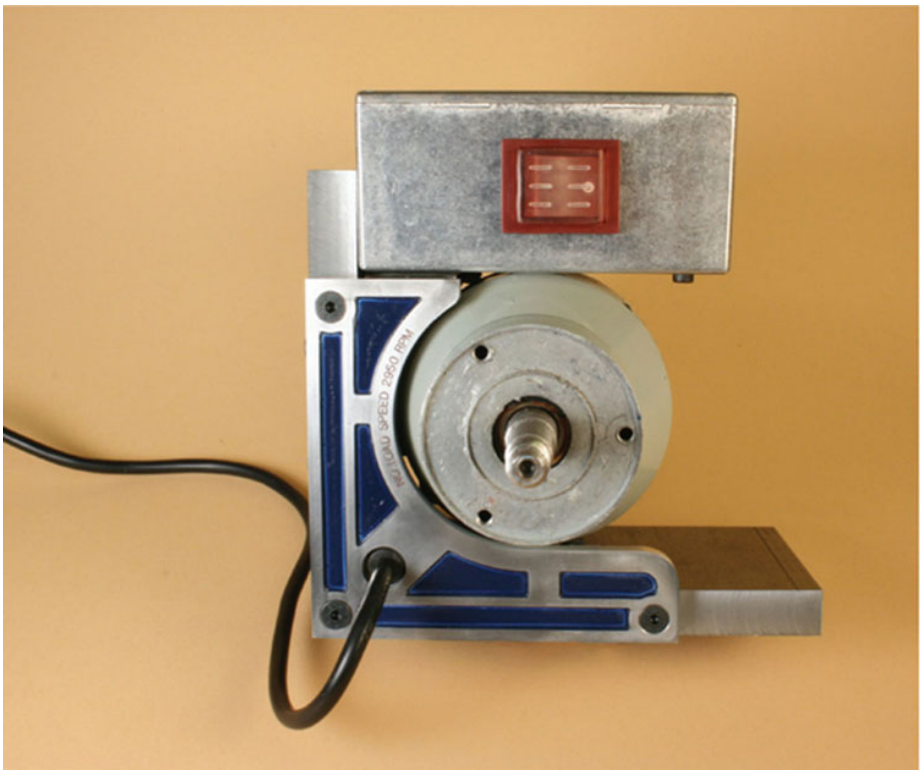


Fig. 1-8 The enhanced bracket in use on a toolpost grinder (note: wheel and guard removed for clarity).

2D design software has been in use since the early 1970s and ranges from simple freeware to powerful industry-standard proprietary packages. While there are many design packages, there are some important factors in the choice of package to use. In a CAD/CAM/CNC system, each stage in the process must be able to communicate with the next stage and this guides the choice of packages.

AutoCAD was an early entrant to the 2D design market on personal computers and its native DWG file format has become one of the standards for file interchange for drawings being passed between CAD and CAM systems. AutoCAD's DXF file format (the Drawing eXchange Format) is another widely used file format for communicating data between the CAD and CAM stages of the CAD/CAM/CNC cycle.

These two file formats are widely recognized by all the useful drawing programs, to the extent that any drawing program not supporting these formats is not likely to be of use in the drawing office or the workshop.

Fortunately, a number of programs support these formats, so the choice is wide. Today, AutoCAD LT continues to be popular, but TurboCAD, DraftSight (free), and many others are all capable of producing 2D DXF drawing files.

2D drawing packages deal with flat shapes, such as parts made from sheet metal. 3D drawing packages deal with shapes such as a human head, or all sides of a locomotive smokebox, or the front, rear and sides of a car wheel.

2D shapes are useful, but relatively flat and featureless. True 3D shapes are difficult to hold for machining, especially if there are features to be cut on all sides like a 3D map of the world, for example. So, 2½D is the most useful kind of shape because, although based on a flat surface, the ability to machine down into the surface at any point allows designs for shapes that are like 3D but

without features on the rear. Those shapes can be held on a conventional milling machine and they can be machined from one side, using movements along X, Y and Z axes.

While 2½D shapes are most common, some readily available software allows fully 3D shapes to be made from 2½D shapes by machining one side, then turning over and machining the reverse side so that the basic 2½D machining movements produce a fully 3D object.

Cut2D

The Vectric Cut2D package is one example of a CAD/CAM package that will allow simple 2½D machining with relative ease. It takes a DXF drawing, or allows the user to draw shapes within the program itself, then outputs the code required for separate CNC software to control a machine. Although there is more work required from the user in dealing with a 2½D object, the way in which the software does its job is essentially the same, whether the object is 2D or 2½D. The limitations on the program are that machined areas should generally have a flat bottom or that the shape of the bottom is formed by the shape of the end of the cutter (for example, a ball-nosed cutter producing a rounded groove).

In the CAD/CAM/CNC sequence, both the CAD design stage and the CAM stage using Cut2D are relatively straightforward. More fully shaped objects, contoured along the Z axis, require more sophisticated CAD software, such as Inventor or Solidworks, and more capable CAM software, such as Cut3D, VCarve Pro, PhotoVCarve, or Aspire.

VCarve Pro

This software package adds the capability of full 2½D machining and a more extended range of drawing tools. The more challenging part of the CAD/CAM/CNC sequence then becomes the CAD first stage,

where the design is conceived, while the CAM stage is relatively easy, compared to the complexity of the design.

Other 2D and 3D CAD and CAD/CAM Software

Typical 2D CAD software includes Draft-Sight, RhinoCAD, AutoCAD and TurboCAD as well as vector drawing packages, such as Adobe Illustrator. Bit-mapped drawing packages, such as Adobe Photoshop, are not suitable for conventional CAD functions, although you will read later about some specialized applications that may benefit from bit-mapped images or drawing software. Typical 3D software includes Inventor, Solidworks and TurboCAD Pro.

There are references to AutoCAD, Turbo-CAD, Adobe Illustrator, Inventor and Solidworks in this book, but other similar packages may be used to achieve the same result.

Preparing for Manufacture

Once a design has been created, it must be translated into a form suitable for computeraided manufacture (CAM). Like the design stage, this can be done manually or with computer assistance. The start of this process is the design, and the end result is data in the form of instructions for a computer-controlled machine tool.

For a simple design, which could be machined using simple movements of a tool between known points, manual coding of the tool movements is perfectly possible. The challenge is in specifying the movements of a tool for a complex shape consisting of multiple curves or surfaces, requiring complex movements of the tool. For that reason, this stage is often carried out wholly or partly by computer. There is a trade-off between time and complexity, and while the tool movements for many designs can be specified by a human, this can quickly become a longer and more difficult task as the complexity of the design increases. Even in a small workshop, time is often a significant factor, and while there is much satisfaction

to be gained from hand coding, CAM software can save time and allows more complex designs to be manufactured. Ideally, both hand coding and CAM software should be available, as a knowledge of hand coding is a valuable skill that allows CAM software to be used efficiently and intelligently in turning a design into data for a machine tool.

CAD and CAM are closely related and there are many software packages that perform both functions, differing mainly in the balance they strike between CAD and CAM. Lower-end packages tend to favour the CAM stage, while several of the high-end packages contain complete CAD and CAM tools, allowing complex designs to be created and turned into data for a machine tool. Single-purpose CAD or CAM packages tend to provide much more sophisticated design or CAM environments, but data normally has to be transferred between packages by a human operator.

CAM software ranges from the simple to the sophisticated, and prices reflect the power of the packages. Cut2D, for example, provides basic 2D and 2½D design tools and complete CAM facilities for turning the design into program instructions for a range of CNC machines. VCarve Pro provides much more capable design tools for 2D and 2½D work as well as the CAM facilities. Aspire provides a full 3D low-relief design capability and all the CAM facilities. The price and complexity of the packages reflect their capabilities.

As with CAD software, many other packages are available that perform the same function, and these include single-purpose DXF to G code convertors that take a DXF file generated by a CAD program and output a program to control a CNC machine, but offer no design tools.

Controlling the Machine

The final requirement is for software to control the motors on the machine tool. This is the CNC software, and if it is loaded with the instructions generated by the CAM software, it will use those instructions to carry out a machining sequence to produce a part.

In this book, the LinuxCNC and Mach3 CNC packages will be used to illustrate the process and provide the examples. Aside from minor differences, most other CNC software packages perform the same function, so that the examples should all be capable of producing the same machining sequence, irrespective of the CNC package used. The only change might be minor amendments to suit the specific requirements of packages provided by other manufacturers or software writers.

Both the LinuxCNC and Mach3 software packages provide a user-friendly environment for controlling a machine and allow direct manual control as well as accepting data from CAM software.

Mach3 runs on computers using the Windows operating system. The specific requirements to allow Mach3 to be installed and run on a computer can be found on the Mach3 website at www.mach3support.com.

LinuxCNC runs on computers using the Linux operating system and, in particular, specific versions of Ubuntu Linux. The specific requirements to allow LinuxCNC to be installed and run on a computer can be found at www.linuxcnc.org.

CNC software also provides facilities for common preparatory tasks such as homing axes, creating soft limits on slide travel, and defining coordinate systems in preparation for machining.

As with CAD and CAM packages, CNC control software is sometimes integrated with CAM or CAD/ CAM software as part of a multifunction software package. Fully integrated CAD/CAM/CNC packages are normally specific to a particular machine or range of machines by a specific manufacturer.

CNC software interprets instructions and sends out signals to control the motors attached to a machine tool. The motors turn feedscrews that move the slides or turn the spindle of the machine.

The beauty of this system is that using the same instructions will cause the same movements of the slides and spindle, so producing the same machining sequence time and time again.

MECHANICS

Typical CNC milling machines closely resemble their conventional manually operated equivalents and have slideways and tool holders, spindles and tables, so the movements made by the various slides is very similar, whether the machine is operated by hand or by CNC.

Fig. 1-9 shows a typical small CNC milling machine in which the manual controls have been replaced by computer-controlled motors that operate the slides.

Fig. 1-10 shows a somewhat larger mill with more powerful motors and larger table movements, but still based on the same principles by which a table moves side to side and back to front, and the head with the spindle, which holds the tool, moves up and down.

Fig. 1-9 A typical small CNC mill from Sherline.

Fig. 1-10 A larger CNC mill from Tormach. (Photo: Tormach LLC)

Some CNC machine designs have evolved to suit particular machining techniques or because of the size or shape of the material being machined. Woodworkers using mainly large sheets of flat material like particle board, for example, often use gantry mills in which the table or bed of the machine does not move, but a router is held in an overhead mount that can move horizontally above the work and vertically away from or towards the work. Fig. 1-11 shows an example of this kind of machine. Although the movements are different, the fact that the cutter moves instead of the table makes little difference to the way in which this type of machine is controlled.

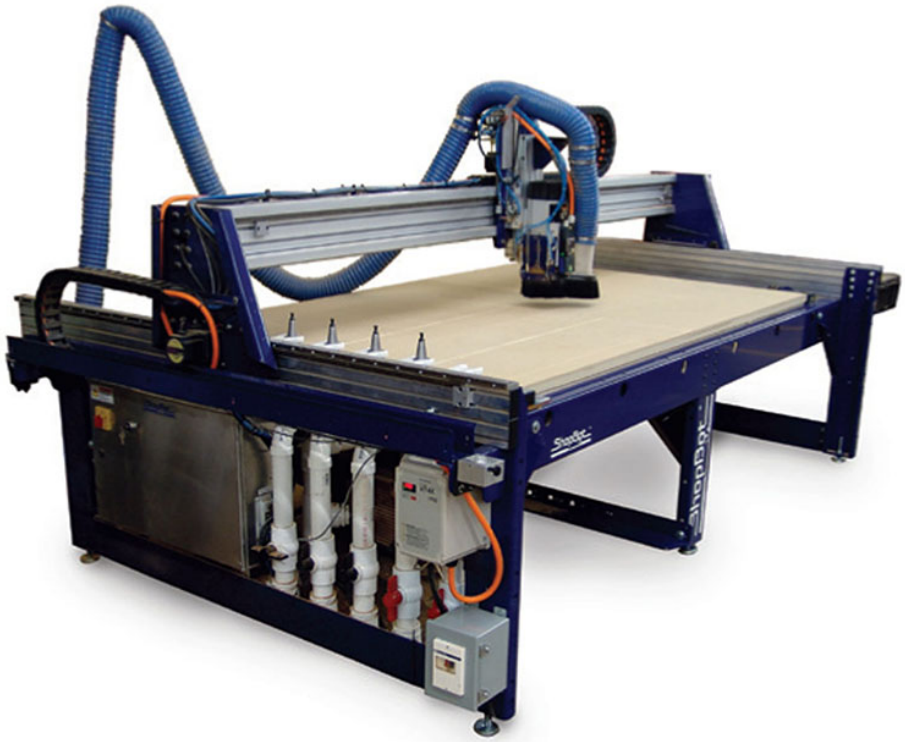


Fig. 1-11 A ShopBot gantry mill. (Photo: ShopBot Tools Inc.)

All CNC machines feature mechanics driven by motors or actuators of some sort, and there are two basic types of motor available for powering slides and controlling their movement. The first is the stepper motor and the second is the servo system.

When a stepper motor receives a signal, its shaft will rotate through a small angle. The motor will then hold its position until it receives another signal. A typical stepper motor used on a machine tool will rotate through 1.8 degrees when it receives a signal, so it will require 200 signals and make 200 steps as it makes one complete turn. Stepper motors used in other applications may make fewer steps every turn, and small stepper motors used in some printers will turn through 7.5 degrees every time they receive a signal, so requiring only 48 signals and making only 48 steps in one complete revolution. The more steps required per revolution, the finer the control of whatever is attached to the stepper.

A servo produces much the same result, but uses a different technique involving continuous error correction. The motor may be one of a range of types, but is capable of rotating, and driving a slide. When the motor controller tries to move the servo (and the slide) to a particular position, the servo begins to turn. On the end of the motor shaft there is a device that will send an error signal if the position of the motor is not where it needs to be for the slide to be in the correct position. This device is most commonly an encoder or a resolver. The error signals are used to correct the movement of the motor until the slide has reached the required position. In this way a servo system incorporates feedback and that is what makes it different to a stepper motor system.

When a signal is sent to a stepper motor, the motor should move. If it is obstructed and is prevented from turning, the control software will not be able to detect that it has not been able to move as intended.

When a signal is sent to a servo, the motor should move. If it is obstructed, the error signals will continue to try to make the servo achieve its target position and the control software will receive signals indicating that the servo is not in the correct position.

It is possible to add a resolver or encoder to the end of the shaft of a stepper motor, effectively turning it into a servo system, but this adds to the cost of the system and the complexity of the electronics and, in cases where feedback is essential, it may be simpler and more effective to use a servo system instead.

At the present time, most small CNC machines for the hobbyist use stepper motors and, where these are of reasonable power, the system not only works well but is cost-effective. Industrial machinery normally uses servo systems, which are both powerful and expensive.

Exactly the same principles of operation apply to both stepper motors and servos, and similar software can be used in most cases. The range of stepper motors commonly available at a reasonable cost includes steppers that are sufficiently powerful to reliably move the slides on a machine as large as a Bridgeport manual mill. On the other hand, where precise control of position is essential, even on a

small machine, servo systems are available in a range of sizes and powers.

The feedscrews on a CNC machine tool may be of conventional screw form, like the Acme screw form (Fig. 1-12), or ball screws and nuts (Fig. 1-13). In either case, when the feedscrew turns, a nut attached to a slide is pushed or pulled to make the slide move. In the ball screw, the nut contains small balls that roll around the surface of the screw and this results in much lower friction compared to the conventional screw in which the flanks of the screw and the nut rub together, because rolling friction is less than the friction between sliding surfaces. But while the ball screw leads to lower friction and requires less powerful motors to move a slide, the main benefit is in the more predictable wear characteristics of the screw. Whereas a conventional leadscrew has a tendency to wear unevenly, ball screws are much less prone to this problem and wear in a more even fashion.

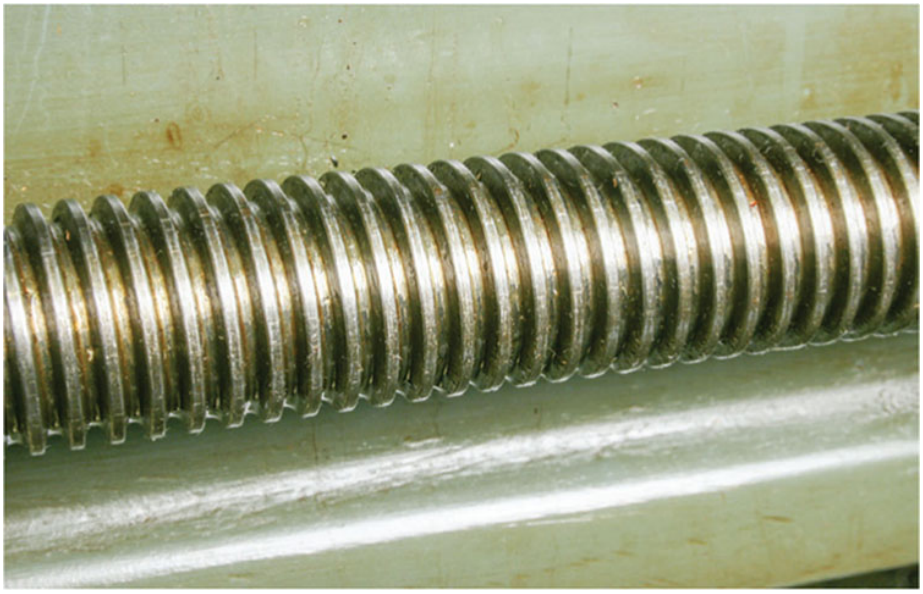


Fig. 1-12 An ACME leadscrew.

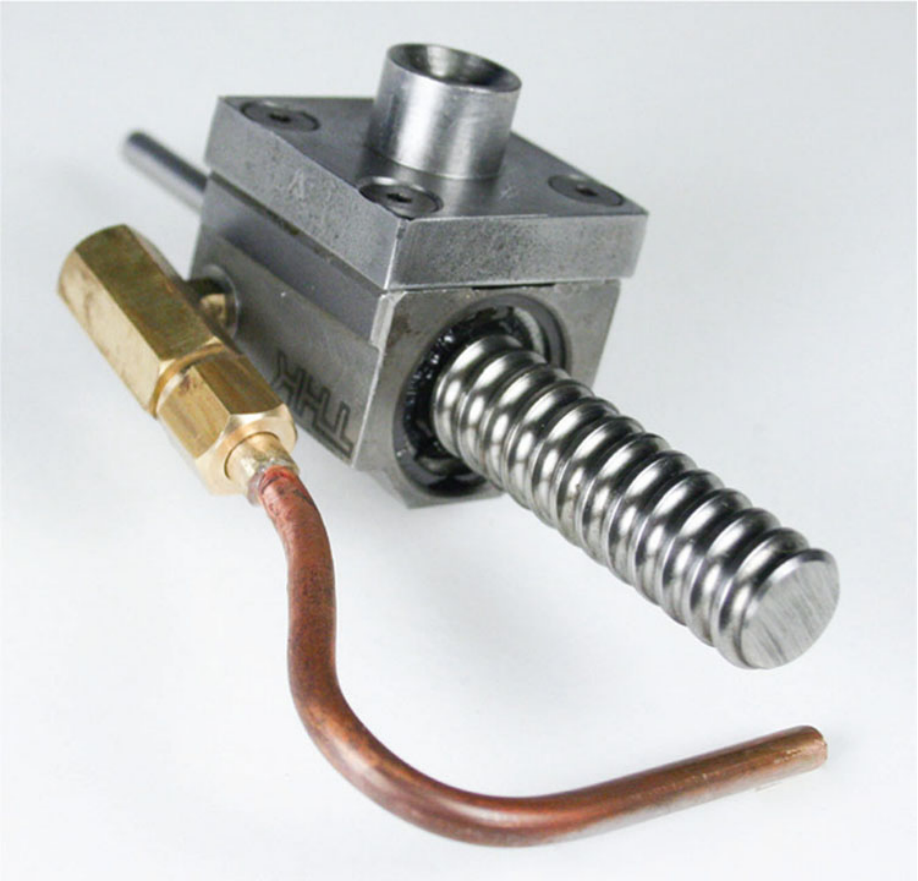


Fig. 1-13 A section of a ball screw and ball nut.

Wear in a leadscrew results in backlash, so that reversing the direction of rotation of the leadscrew results in a partial turn that causes no movement because the flanks of the screw take up the slack within the nut before pushing it in the opposite direction. This kind of lost motion of the nut and the slide results in steps moved by the stepper motors or servo system with no corresponding movement of the slide. The slide and the motors get out of sync both with each other and with the controlling software, which leads to errors in the machining path.

Servo systems may be able to cope with lost motion, especially if they are referencing to a scale mounted on the slide itself, but stepper systems cannot make up for lost steps.

The most popular software packages have a system for specifying the backlash in a leadscrew and can then compensate by recognizing the need for some extra steps when the direction of a

slide changes and there is the potential for lost motion. What is more difficult is for software to compensate for backlash that is different at different points along the length of the leadscrew. So the next best thing to no backlash at all is uniform backlash, while the worst situation is a leadscrew that has significant backlash that varies from point to point along the leadscrew. Although ball screws tend to have a more consistent degree of backlash along their length, they are not without their own drawbacks. Backlash can be eliminated by using ground ball screws and preloaded nuts. These are expensive and the ball screws usually have to have a large diameter (16mm or more).

The low friction of a ball screw is not always an advantage and where there is backlash, the low friction can lead to the cutter snatching as the slide changes direction. The cutter grabs the work and moves the slide, ruining the machining pattern and perhaps damaging the workpiece and cutter. As with most engineering problems, there is a compromise between the ideal design and real-world reality.

Until recent times, most machine tools had dovetail slides (as on most mill tables) as can be seen at the front of the cross slide in Fig. 1-14. In an effort to reduce friction, manufacturers have adopted slideways based on the same rolling-ball technology used in ball screws, and modern large machine tools often use ball slides where the conventional mating surfaces are replaced by rails or rods on which roll tubular or rectangular nuts containing balls (Fig. 1-15). As with ball screws, rolling friction is less than sliding friction, so less power is needed to move the slides. Smaller machines have been slower to adopt this technology, simply because of the expense, but this is a design feature that is gradually being adopted in more expensive machine tools. It is also found on some older training machines that are now becoming available on the second-hand market.

Fig. 1-14 A dovetail slideway on a cross slide.

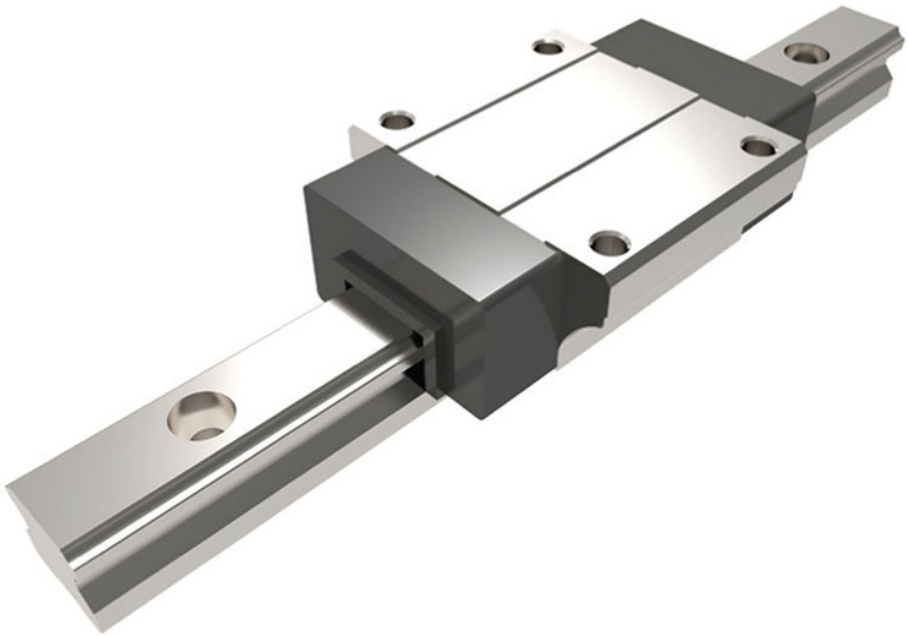


Fig. 1-15 Linear guide with rail and carriage. (Photo: Automotion Components Ltd.)

Stepper motors and servos may be directly connected to feedscrews or they may be connected via pulleys and belts. The pulleys must be coupled together in such a way that there is no lost

motion between them as they turn, so that the pulley system does not introduce backlash. This is usually done by using toothed belts that contain a core of steel wire. There are different proprietary shapes for the teeth of these pulleys, but as long as the same make of pulleys and belts is used, this should cause little trouble.

Spindle motors may be free-running or under computer control. In most small machines, the speed of the spindle motor is not controlled by the CNC software. It may be switched on manually or the CNC software may switch the motor on but not control its speed. Ideally, the CNC software should be able to control the position of the spindle as it rotates, or at least sense its rotations, so that the rotation of the spindle can be coordinated with the movement of the various slides, perhaps for thread cutting or some spiral machining operations. At the present time, not all small CNC machines have spindle speed control or spindle position sensing, which places some limitations on operations like tapping, machining threads and dividing using the main spindle.

ELECTRONICS

Fig. 1-16 shows some of the electronics forming a typical CNC control system. Details will vary from one system to another, but the core functions will be present in most machines.

Neither stepper motors nor servo systems are connected directly to the computer, because the motors require considerably more power than the computer could supply. Instead, the computer sends low-powered logic-level signals to a driver unit, which responds to the signals by generating pulses of much higher voltage and current to suit the motors. Typical voltages range from 5 to 70 volts or more, and typical currents depend on the rating of the motors, with some powerful stepper motors requiring up to 10 amps at 70 volts for each motor.

To protect the computer system, an interface board is often used to buffer or isolate signals passing from the computer to the motor drivers, so that there is no possibility of high voltage or current

flowing from the motor drivers back into the computer. Isolation may be by opto-couplers that effectively transmit signals with no electrical connection between the computer side and motor side of the interface board.

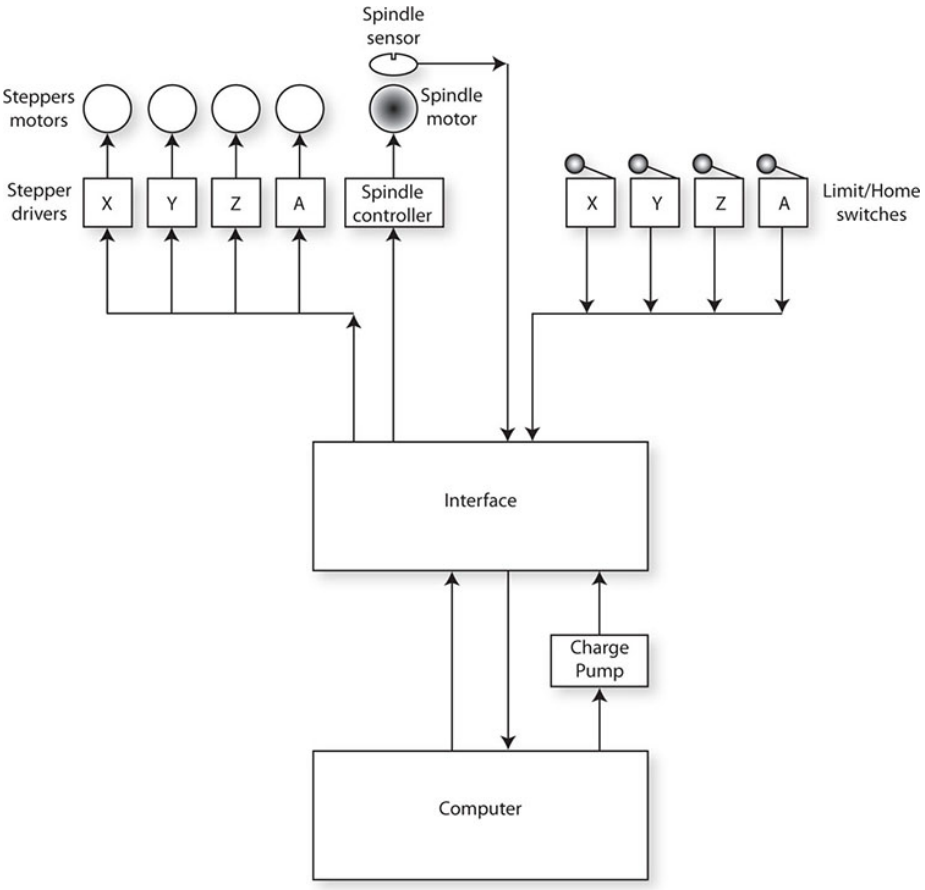


Fig. 1-16 Schematic of typical electronic items associated with a CNC milling machine.

A CNC machine may incorporate sensors designed to indicate when slides reach particular positions (**limit** or **home switches**, for example) or the speed of rotation of a spindle. These sensors send signals to the computer system, but normally pass through an interface first where they are turned into a form suitable for the computer, with appropriate voltage levels or data formats. As with the signals to the motor drivers, the interface provides isolation between the sensors and the computer.

Because a machine tool is a potentially dangerous device, additional safeguards may include a charge pump that allows the hardware to operate provided the charge pump receives a constant set of signals from the control computer. If the CNC control software stops running or a fault develops in the computer, the charge pump will stop receiving the stream of signals from the computer and will disable the signals to the hardware to stop the machine.

On top of all of this, there may be an emergency manual cut-out switch for the spindle motor of the whole system, and the computer software can also be stopped by pressing an appropriate key on the keyboard.

Finding the shape of holes in an engine block using a probe. (Photo: Centroid Corporation.)

2 The Controlled Point

In this chapter you will learn about:

- the Controlled Point, the axes of movement, and coordinate systems;
- homing and touching off.

CNC software moves a Controlled Point in relation to the workpiece as it carries out a machining sequence. The Controlled Point may be anywhere, but is normally a reference point at the end of the tool, on the same axis as the spindle, as shown in Fig. 2-1.

It does not matter whether the head moves or the slides move, the software will always act as though it is moving the Controlled Point. It is best to focus on what happens to the Controlled Point, so what really matters is not the way the parts of the machine move, but the way the Controlled Point moves in relation to the work.

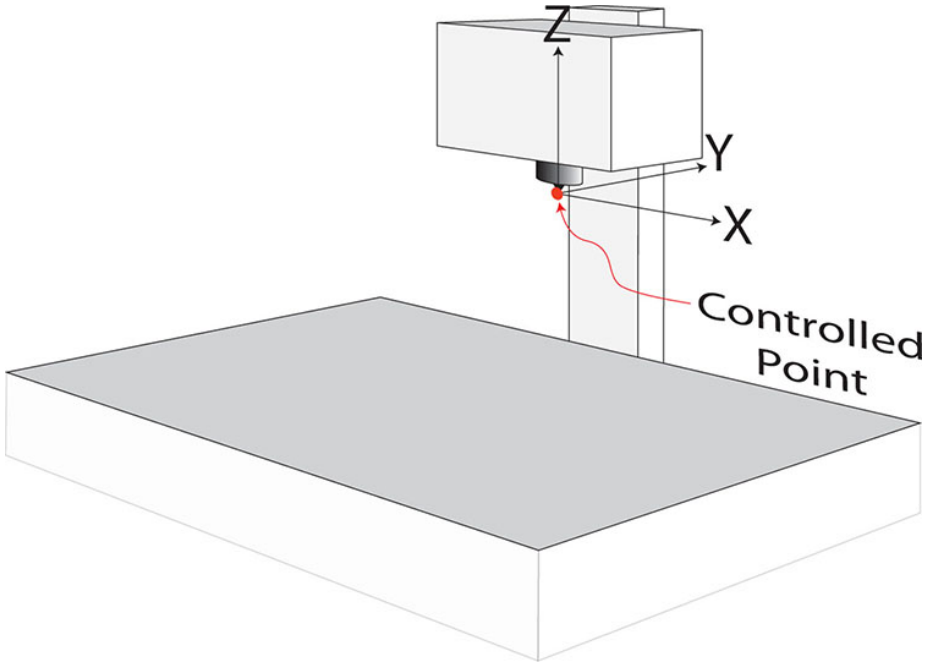


Fig. 2-1 The Controlled Point.

Looking directly down on to the table, the XY plane is parallel to the flat surface of the table, with the X axis running left to right and the Y axis running front to back (Fig. 2-2). The origin is the point where the X and Y axes cross and both axes have a value of zero at that point.

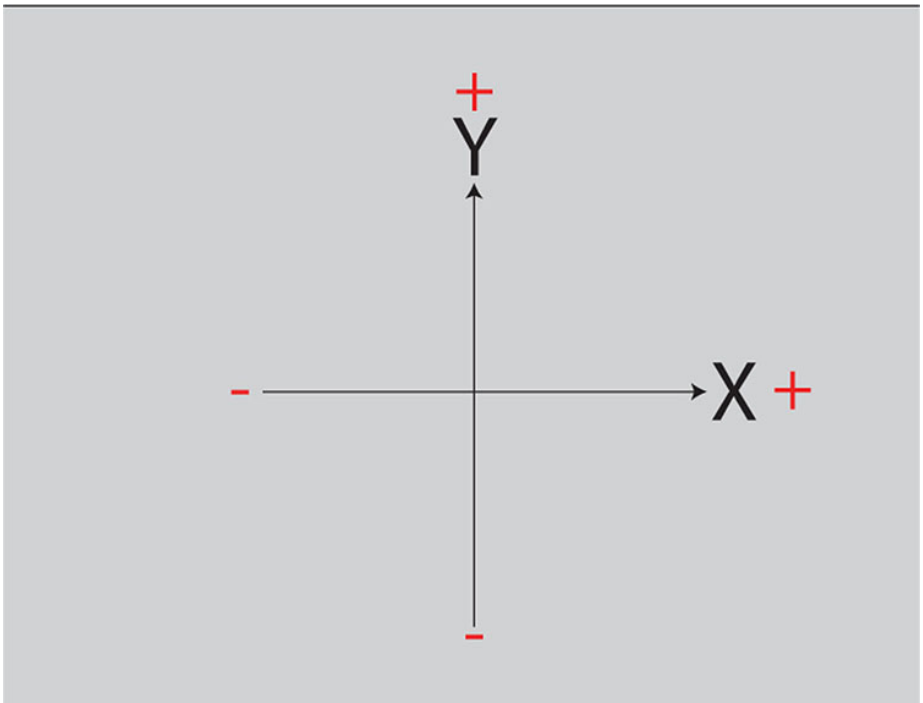


Fig. 2-2 The XY axes and the origin.

On the X axis, the X value becomes more positive as the Controlled Point moves to the right in relation to the table, and more negative (or just less positive) as the Controlled Point moves to the left. The Y value becomes more positive as the Controlled Point moves to the rear.

The Z axis runs vertically up and down at right angles to the XY plane (the table) and represents the vertical movement of the Controlled Point. The Z value becomes more positive as the Controlled Point moves further away from the table.

For a ‘tabletop’ vertical milling machine, in which the head moves up and down and the table is fixed to the cabinet or bench, Fig. 2-3 shows the orientation of the axes for movements parallel to the X axis (left and right), the Y axis (forwards and backwards) and the Z axis (vertically up and down).

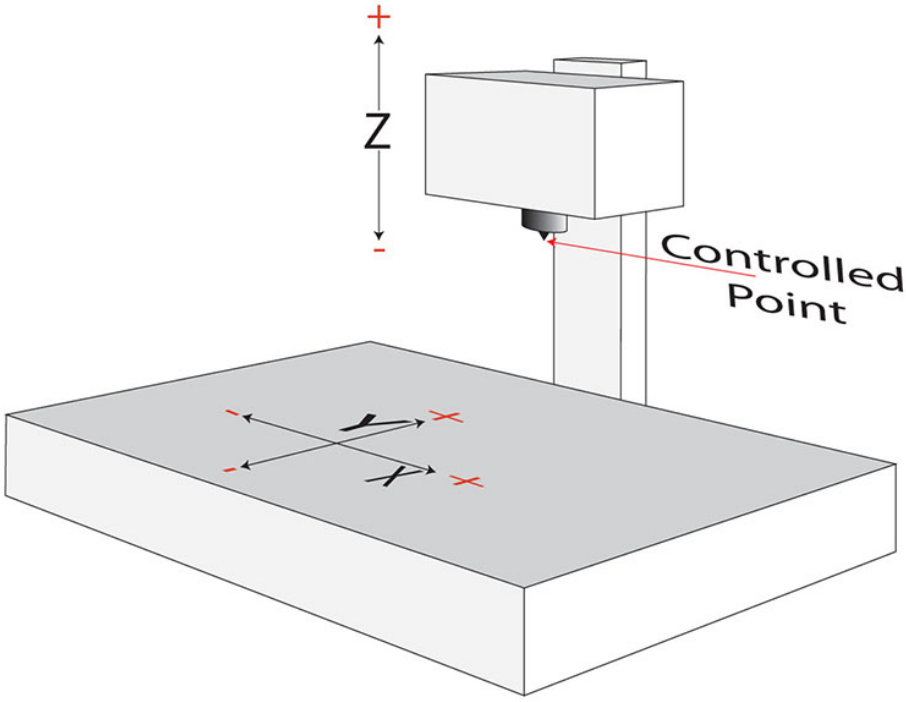


Fig. 2-3 The axes on a tabletop mill.

Because the table remains fixed and the head moves up and down, the Z value becomes more positive as the head is raised and moves the Controlled Point in the positive Z direction and further away from the work.

On a ‘knee’ mill, in which the head is fixed but the knee moves the table up and down, the Z value becomes more positive as the table is lowered, taking the origin with it, and the distance between the Controlled Point and the work increases (Fig. 2-4).

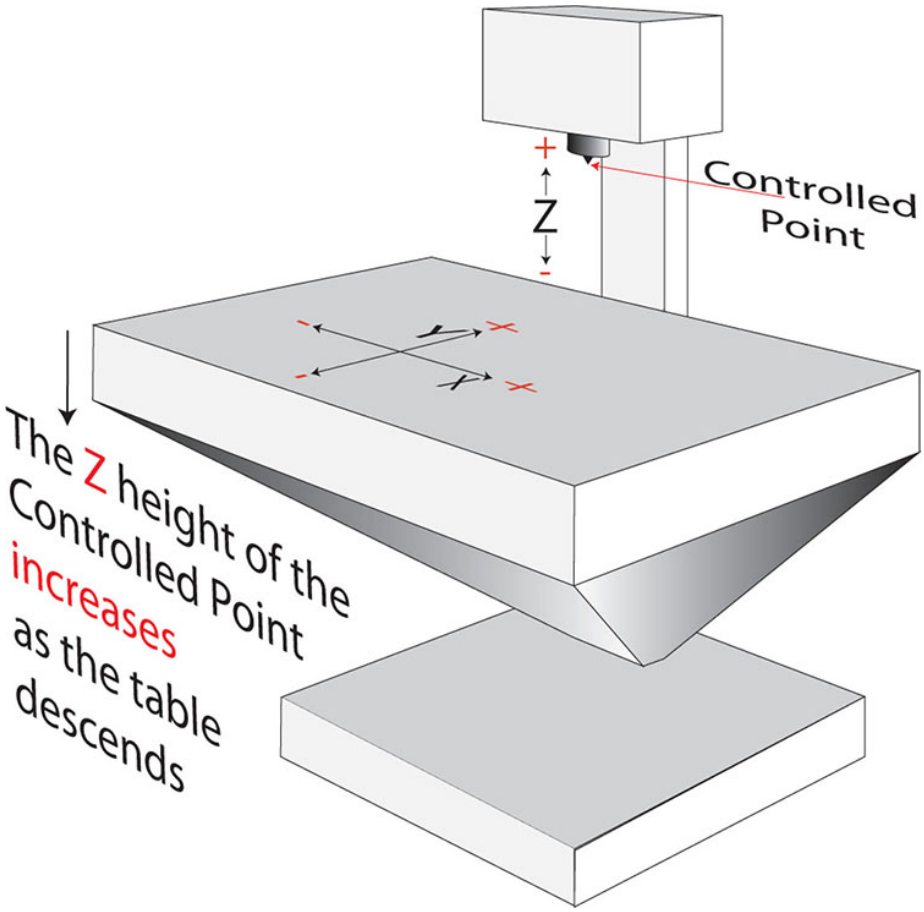


Fig. 2-4 The axes on a knee mill.

On a gantry mill, where the table does not move but the head is mounted on a carriage that can move across the gantry as well as up and down (Fig. 2-5), the Z value becomes more positive as the head moves up, the X value becomes more positive as the gantry moves to the right and the Y value becomes more positive as the carriage moves towards the back of the bed.

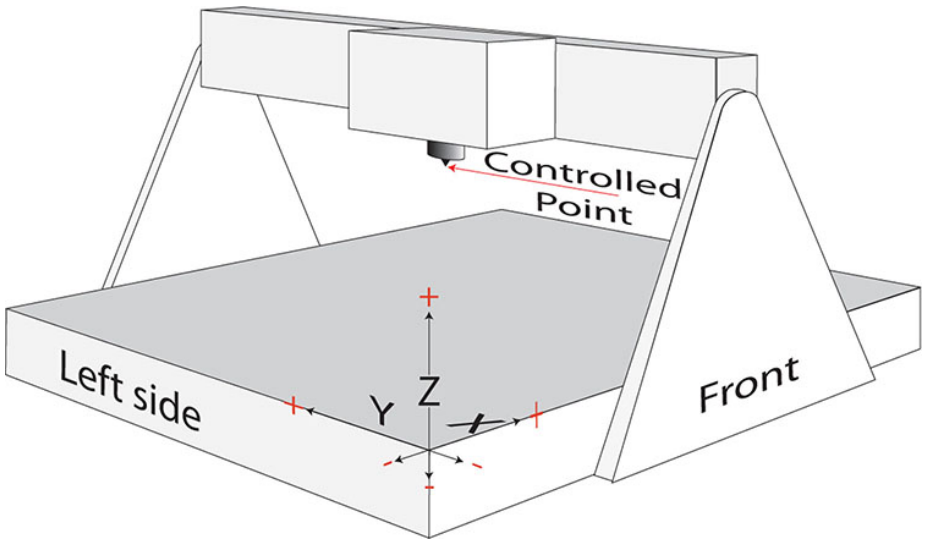


Fig. 2-5 The axes on a gantry mill.

Fig. 2-6 shows the coordinates of some points on the XY plane (the surface of the table). The X value is always written first and the Y value second, and (0,0) is the origin or starting point, so that (4, 2) represents a position 4 units from the origin along the positive direction of the X axis and 2 units from the origin along the positive direction of the Y axis. Negative values represent distances in the other direction, so that (−5, 2) represents a point 5 units from the origin along the X axis in the negative direction and 2 units from the origin along the Y axis in the positive direction. (2, −3) is 2 units from the origin along the X axis in the positive direction and 3 units from the origin along the Y axis in the negative direction.

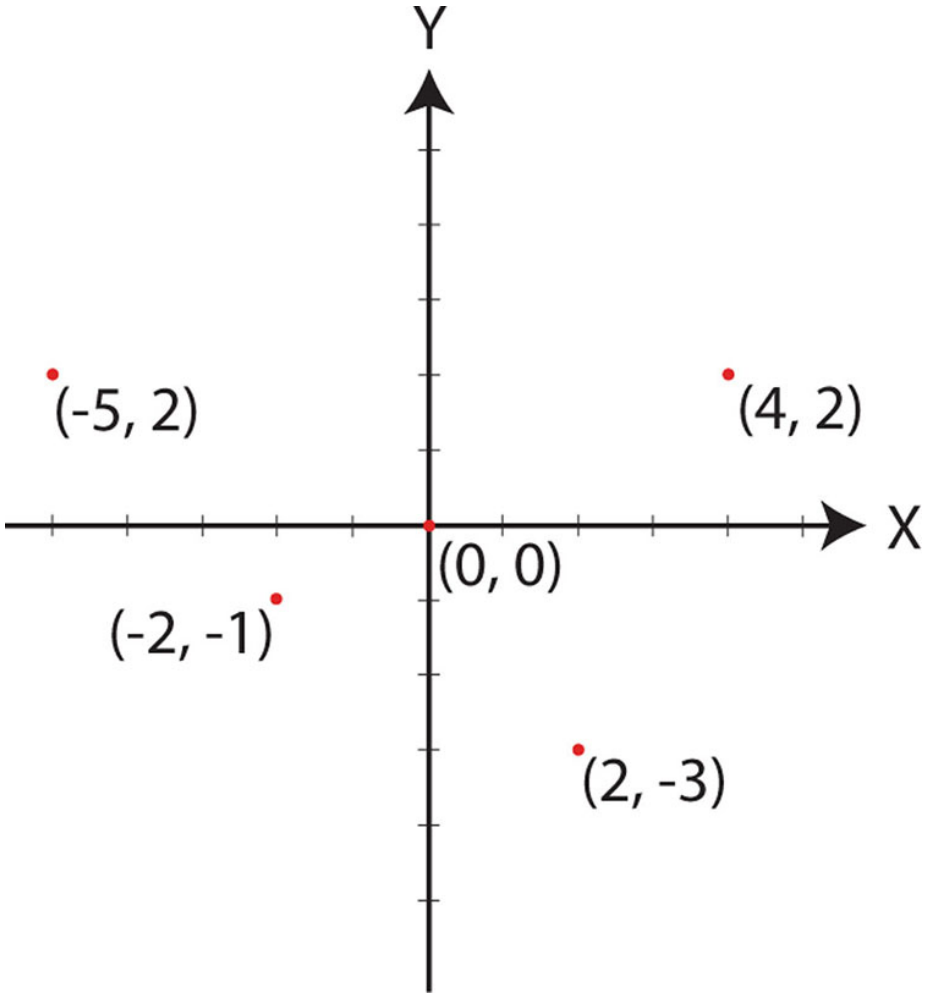


Fig. 2-6 Some examples of coordinates.

Positions of the Controlled Point usually have three coordinate values (an X, a Y and a Z value) so that the software can keep track of the position of the Controlled Point in three-dimensional space.

So-called 2D machining takes place with the Controlled Point parallel to the XY plane or, less usually, the XZ plane or the YZ plane. 3D machining takes place in three dimensions within the boundaries of the whole space inside which the Controlled Point can move. These definitions are a little imprecise, though.

Moving the Controlled Point in only two dimensions would mean that every cut would have the same depth. Moving the Controlled Point in three dimensions to access any point in space would be impossible with a real workpiece because if the mill attempted to

machine the back surface of a workpiece, it would hit anything that existed above the cutter.

A better description of real-life machining on a conventional mill or lathe would be 2½D, where the Controlled Point travels parallel to the table but also moves up and down to cut at various depths, with the restriction that the rear surface of the workpiece cannot be cut without turning the work over.

LIMITS OF TRAVEL

In a real machine, there are limits to the distance the Controlled Point can travel in any direction before it meets an obstruction or comes off the end of a leadscrew, and there may be electronic switches to prevent damage to the machine, the tool or the workpiece. These are the limit switches that set the limits of travel for each axis.

Fig. 2-7 shows an electronic limit switch with an optical sensor that can be interrupted by a ‘finger’ attached to a moving slide and Fig. 2-8 shows an electrical limit switch that is operated by pushing a roller. Electronic and electrical switches are designed to cut the power to the axis motor to prevent mechanical damage before the slide or feedscrew hits a physical stop. Hitting a physical stop means something has to give and it is better to avoid damage by stopping the motor before that happens.

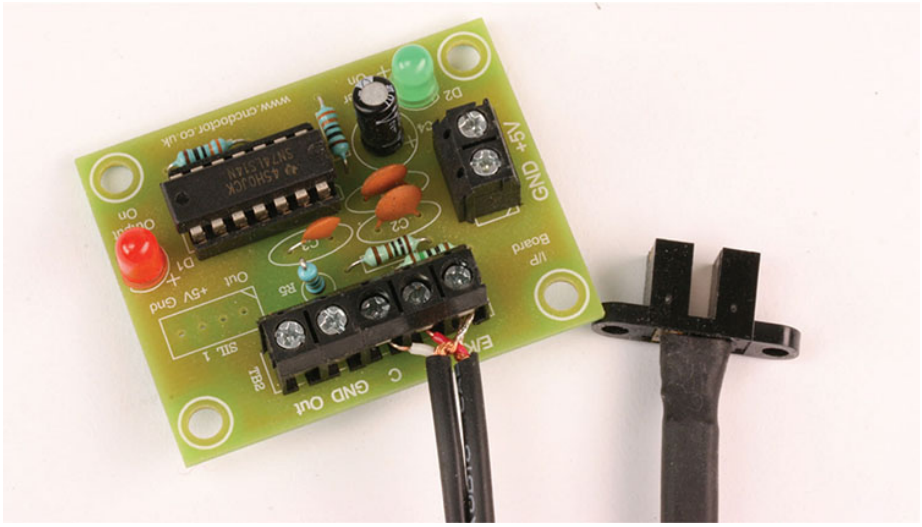


Fig. 2-7 An electronic limit switch.

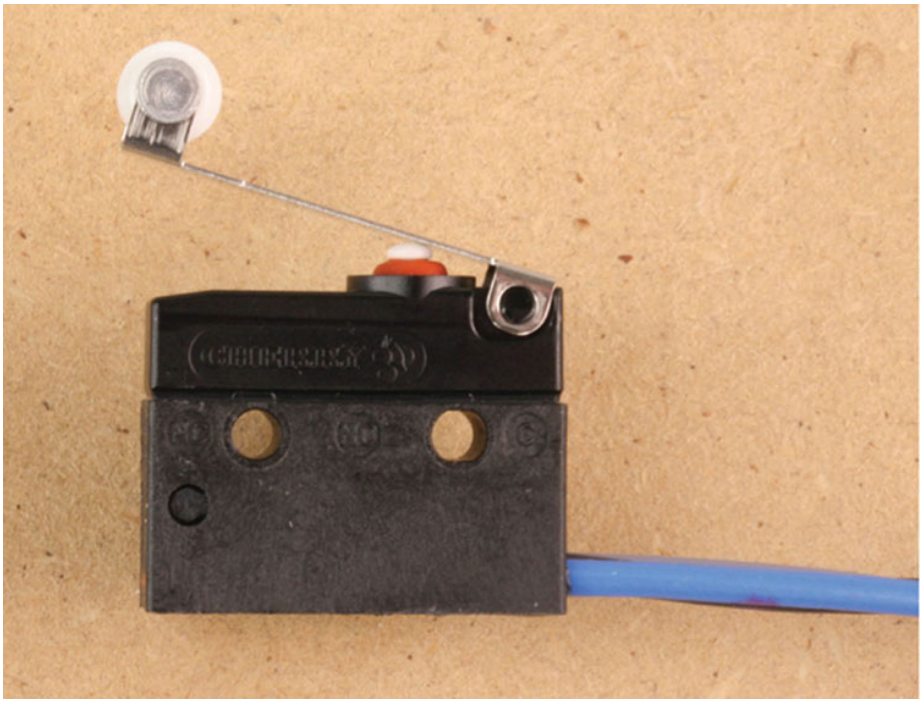


Fig. 2-8 An electrical limit switch operated by depressing a roller on the end of an arm.

Although there are physical limits to the travel possible on a particular machine because of the lengths of the slides and feedscrews, there may also be limits for an individual work set-up, where the travel of the Controlled Point needs to be limited in any

direction to avoid the cutter hitting a clamp or another fixed item like a vice or part of a fixture. In that case, there are soft limits that can be set within the CNC control software. These provide temporary limits on movement, but they are independent of the physical limit switches that are designed to provide overall physical limits on the slide movements. Each linear axis (X, Y and Z) should ideally have two limit switches, one at each end of its travel. That is very often not possible because of limitations on the number of inputs, particularly on machines controlled through a PC printer port. In that case, there should be one limit switch for each axis and a set of soft limits for each axis. The limit switch will provide a way of establishing the Home position for an axis, and the soft limits will prevent the control software from trying to move beyond the physical limits of axis travel. In that case, the limit switches act as home switches and are referred to as home switches in the rest of this book.

HOME AND THE WORK ORIGIN

Once a workpiece is placed on the bed of the milling machine or in the vice, it is essential that the position of the workpiece is accurately known. That way, the Controlled Point can be positioned and moved in relation to the workpiece to machine the various features.

The machine will have a set of **absolute coordinates**, called **machine coordinates**, which will have an origin at (0, 0, 0), known as the Home position. If your mill has home switches, they can be used to set the position of Home. If not, the software will assume some Home position when it first starts up and will display the **current coordinates** of the Controlled Point on the screen. Machines that do have home switches can be made to use a set of moves in a homing sequence to set Home for the machine coordinates by moving towards each home switch until it trips, then backing off slightly.

Mach3: the Ref All button makes the machine move its axes towards the home switches, stopping when those switches trip.

LinuxCNC: the Home All button homes the selected axis. If that button is marked Home Axis, it is because not all axes have homing sequences set up.

If a homing sequence is set up: we need to be able to set the origin of the workpiece, then work from there using coordinates that are related to the origin of the workpiece. This makes it much easier to work out the movements of the Controlled Point in relation to the workpiece. The work origin would normally coincide with the origin set in the CAD program when designing the features to be machined in the workpiece.

The usual way of dealing with this is to use an offset to define a work origin (0, 0, 0) in a known position in relation to the workpiece (Fig. 2-9). Often, this is the front left corner of the workpiece, where it is convenient to have the work origin of the X and Y axes. X values for the movement of the Controlled Point will be positive from the origin to the right and negative from the origin to the left. Y values will be positive towards the back of the workpiece and negative from the origin towards the front. Usually, the Z origin is set to the top surface of the workpiece, although this may vary, depending on the nature of the work and the machining operations to be carried out.

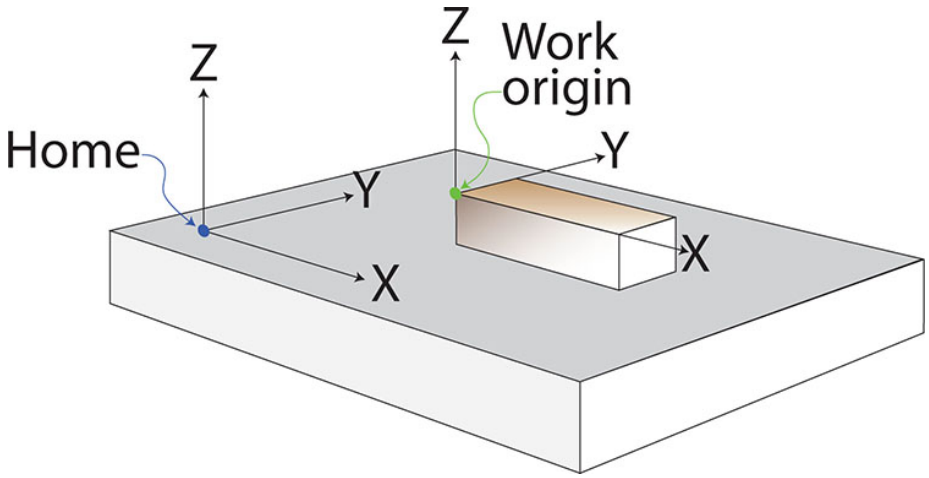


Fig. 2-9 The Home position and a work origin.

Once the work origin has been defined, we can work with a set of coordinates that will give accurate locations in relation to the workpiece, and it is easy to move the Controlled Point to appropriate locations to carry out machining and create the various features on the finished workpiece.

The procedure is quite simple, but it does depend on being able to locate the edges and surfaces of the workpiece accurately in the first place. There are several methods of doing this, depending on the accuracy required.

You should also be aware that the CNC software will carry on working from the Home position of each axis, but will add an offset value to those so that you, as a user, see coordinates related to the work origin (often called current coordinates). Most software has a facility for displaying the position in machine coordinates on request, and those may differ from the readings on digital read-outs (DROs) showing the current coordinates measured from the work origin. In all cases, the machine coordinates show distances from the Home position, while the current coordinates show distances measured from the work origin. Once a work origin is set, there is little need to see or display the machine coordinates. However, although offsets are very useful and very necessary for most jobs, there are some consequences you need to be aware of if you are using the more

advanced features associated with offsets. These are dealt with later in this book.

Setting the work origin uses a procedure called **touching off**, which may have its origins in the physical procedure for locating the work origin by touching the workpiece with a probe. Nowadays, not all probes need to touch the work to locate a position, but the term has stuck. When you touch off, you define an offset that will set the work origin for an axis. This is done by entering a value representing the position of the Controlled Point in relation to the work origin.

Most CNC software will allow you to view the offsets currently in use. Most CNC software also allows you to select any of several coordinate systems to be the current coordinate system and that has some real uses, as explained in Chapter 8. For the moment, it is enough to know that touching off is used to set the origin of the work coordinates in relation to the workpiece by using what are termed G54 or Fixture 1 offsets. When you set a work offset, you are setting the G54 offset, unless you instruct the software otherwise.

TECHNIQUES FOR SETTING THE WORK ORIGIN

How you set the work origin depends to a large extent on how accurately it needs to be set for any particular job. It also depends on the features of the CNC software being used.

If, for example, a small finished object is to be produced by cutting it from a large sheet, there will be enough leeway in the waste around the finished object for a simple pointer, aligned by sight, to be sufficiently accurate for identifying a position where the origin could be located. If, however, an object is to have additional machining done in a precise relationship with other existing faces or features, aligning by sight is unlikely to be sufficiently accurate and more sophisticated methods must be used to locate the origin.

Using a Wiggler

A wiggler is a simple device, consisting of a body and a probe arm, used to locate the vertical faces of a workpiece (Fig. 2-10). The probe arm is attached to a ball that sits inside the body so that the arm can move laterally. The outer end of the arm carries a ball or a cylinder of known diameter.

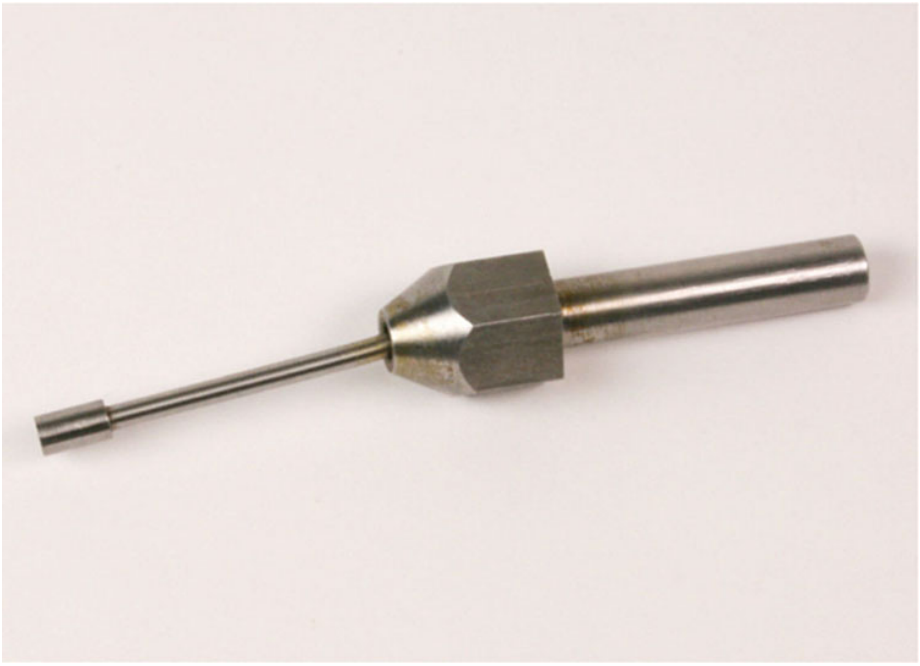


Fig. 2-10 A wiggler.

To touch off the X axis, the probe should be located to the left of the left edge of the workpiece and movements should take place along the X axis. The probe is held in the spindle and rotates with the spindle at a low speed (say 500rpm), the end wobbling about as it does so. Moving the Controlled Point so that the outer end of the arm is slowly brought into contact with the leftmost face of the workpiece makes the outer end run in a more circular path until it is finally concentric with the axis of the spindle, such that the end turns but does not wobble. Advancing the Controlled Point further in the same direction makes the arm flick out of alignment and run along the face of the object, visually indicating that the Controlled Point has passed the point at which the arm can rotate concentrically (Figs 2-11, 2-12 and 2-13 show this sequence). At that moment, the

Controlled Point is half the diameter of the end of the arm away from the face of the workpiece. If the arm carries a 6mm (¼in) diameter cylinder, the Controlled Point will be 3mm (1/8in) away from the face. Because the probe is to the left of the intended work origin, it will be at a negative X coordinate, i.e. −3mm or −0.125 in.

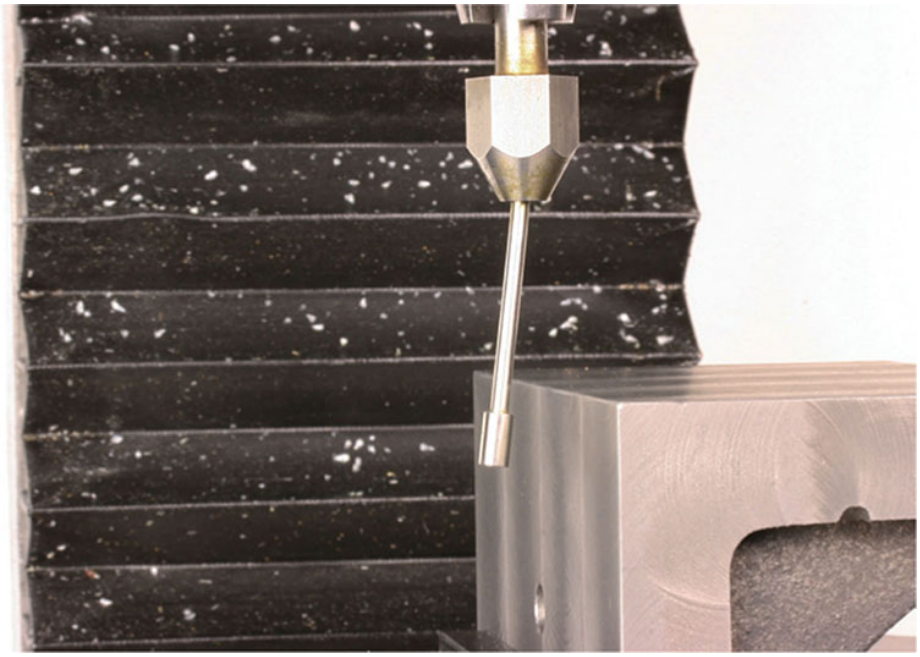


Fig. 2-11 A wiggler about to contact a workpiece.

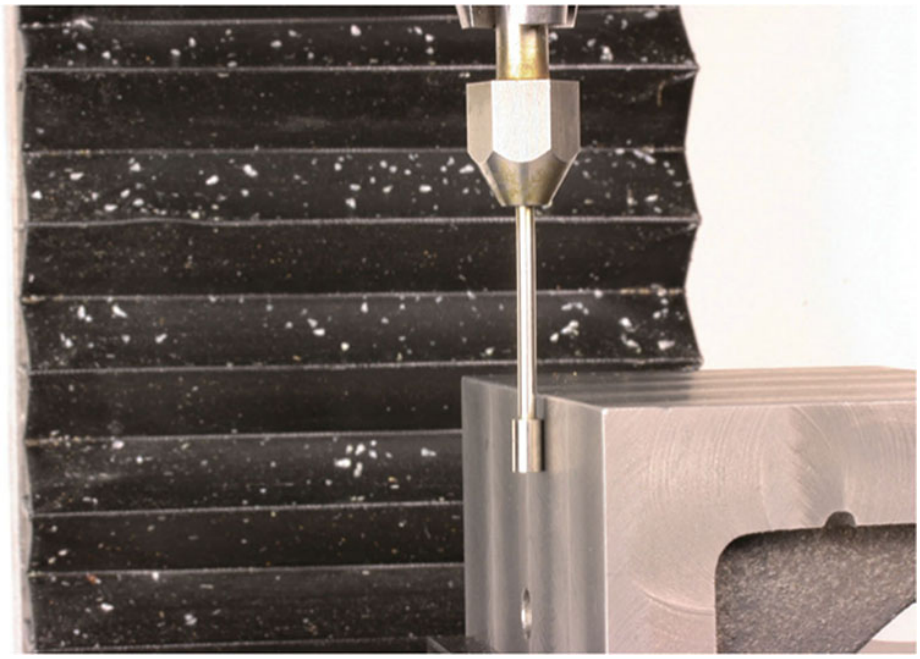


Fig. 2-12 A wiggler arm rotating concentrically with the spindle axis.

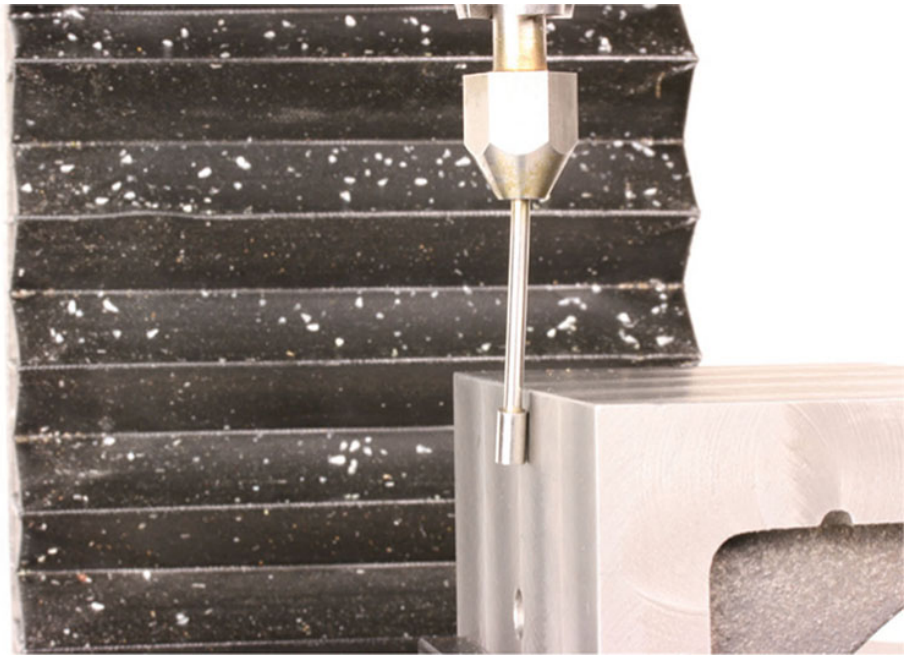


Fig. 2-13 A wiggler arm no longer rotating concentrically with the axis of the spindle.

Mach3: At the Program Run screen, click Zero X, and make sure the Machine Coords button is not lit. Go to the Offsets screen and under Active Work Offset select Fixture 1 (G54). In the section entitled Please Select Edge Finder location, enter the diameter of the head of the wiggler (6mm or 0.25in), then click the Select button for the axis you are touching off (but note that the bottom left button selects the X axis, while the bottom right button selects the Y axis).

Go back to the Program Run screen and you should see the X-axis read-out now displays −3 (or −0.125) because that is the current position of the Controlled Point in relation to the edge of the workpiece.

LinuxCNC: At the main screen, select the X-axis then click the Touch Off button. Make sure the offset you are using is G54 offset 1, and enter a negative value of half the diameter of the wiggler (-3mm or −0.125in) to indicate that the Controlled Point is currently located that distance to the left of the intended position of the work origin.

The Y axis origin can be set using a similar method, beginning with the probe to the front of the front face of the workpiece and advancing along the Y axis towards the workpiece. Wigglers cannot be used to set the origin of the Z axis.

How accurately can a wiggler locate a face? That depends on the friction in the ball of the wiggler and on human judgement as to when the wiggler flicks off from a concentric rotation. With very small movements, there is a judgement to be made. Decently made wigglers should be accurate within 0.02mm (0.001in) or so and, perhaps more importantly, should be repeatable.

Using an Electronic Edge Finder

Electronic edge finders are probes that light up when they touch a metal surface. The probe is mounted in the spindle, like a wiggler, but does not turn. The arm of the probe is rigid, except for a spring-loaded ball at the end. The probe body contains a battery and a light. The probe body, the battery, the light and the ball form a circuit, but the ball is insulated from the probe body and the circuit is not completed until the probe touches the face of a conductive (usually metal) workpiece. At that point, the circuit is completed through the workpiece and the metal parts of the milling machine, so advancing the probe towards a metal face makes the probe light up when it touches the face (Fig. 2-14). The probe is now half the diameter of the tip away from the face, just like a wiggler. Because the probe is on the left of where we want the work origin to be, enter the X coordinate of where the Controlled Point is in relation to the work origin, that is, half the diameter of the tip away, to the left. If the tip is 5mm (0.2in) diameter, enter −2.5mm (−0.1in). The value is negative because the Controlled Point is sitting to the left of the intended position of the work origin.

Fig. 2-14 An electronic edge finder indicating that it is in contact with the workpiece.

The purpose of the spring loading is to protect the probe by allowing the end to move slightly rather than bend the arm of the probe. When using the probe, the aim should be to touch the face without moving the ball, so small movements should be used as the probe gets closer to the face. As with a wiggler, good electronic edge finders should be accurate and repeatable.

Using A Machine Scope or Optical Edge Finder

Optical edge finders, or machine scopes, are like microscopes for machine tools (Fig. 2-15). The scope is held in the spindle and the

operator looks through the eyepiece. The scope has a lens that looks vertically down and it carries cross hairs so that the axis of the spindle can be aligned with an edge, a hole or a mark on a workpiece. In use, it is focused on the work by raising or lowering the Z axis, and the scope shown is in focus when it is 50mm above the surface of the workpiece. The X and Y axes are jogged until the edge or other feature of the workpiece is aligned with the cross hairs (Fig. 2-16). The axes can be touched off at that point. For a rectangular workpiece, the X axis is touched off by aligning the cross hairs with the left face and then the Y axis is homed by aligning the cross hairs with the front face.

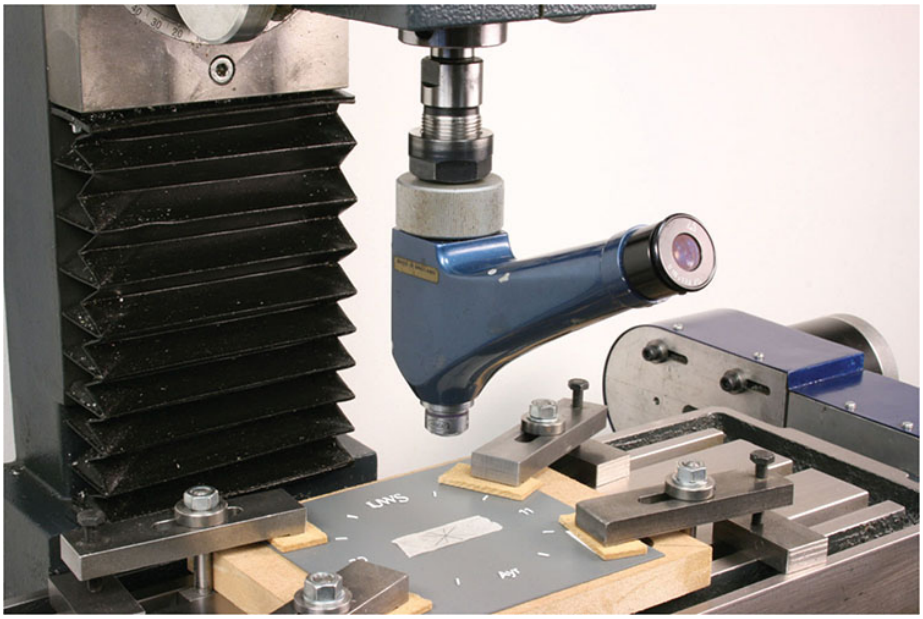


Fig. 2-15 An optical machine scope.

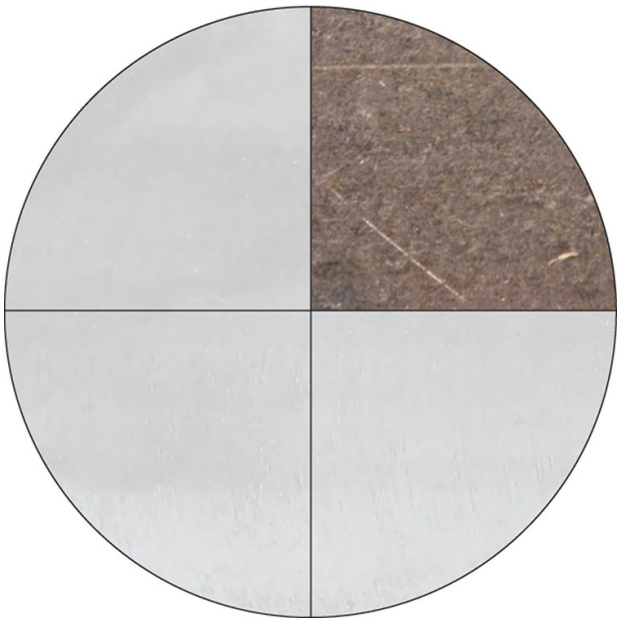


Fig. 2-16 Simulated view provided by an optical machine scope whose axis is aligned with an edge.

Fig. 2-17 shows the cross hairs aligned with the centre of a hole. There is some judgement required in this case, so an alternative is to move the slides so that the cross hairs are aligned with one edge of the hole and touch off there, then move in the same direction to align the cross hairs with the opposite edge of the hole and note the distance as shown on the axis read-out of the CNC program. Halve that distance. Move the slide back to a position well before the first edge of the hole, then forwards to that half-distance point (Fig. 2-18). Touch off again at that point. The movement backwards is designed to counter any backlash in the slide movement so that the various points are approached from the same side, in the same direction. Repeating this sequence for the other axis results in the centre of the hole being at (0, 0). This same sequence of taking half the distance from one edge to another is the basis of movements often used with a probe, as we will see later.

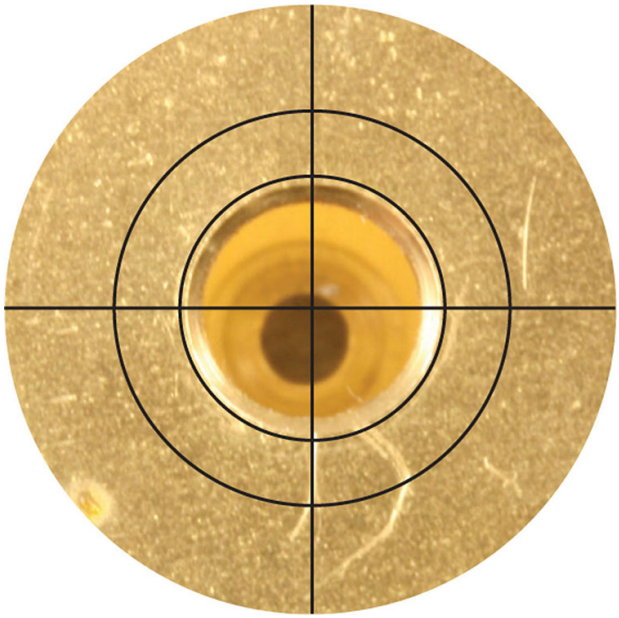


Fig. 2-17 Simulated view provided by an optical machine scope whose axis is aligned with the centre of an existing hole.

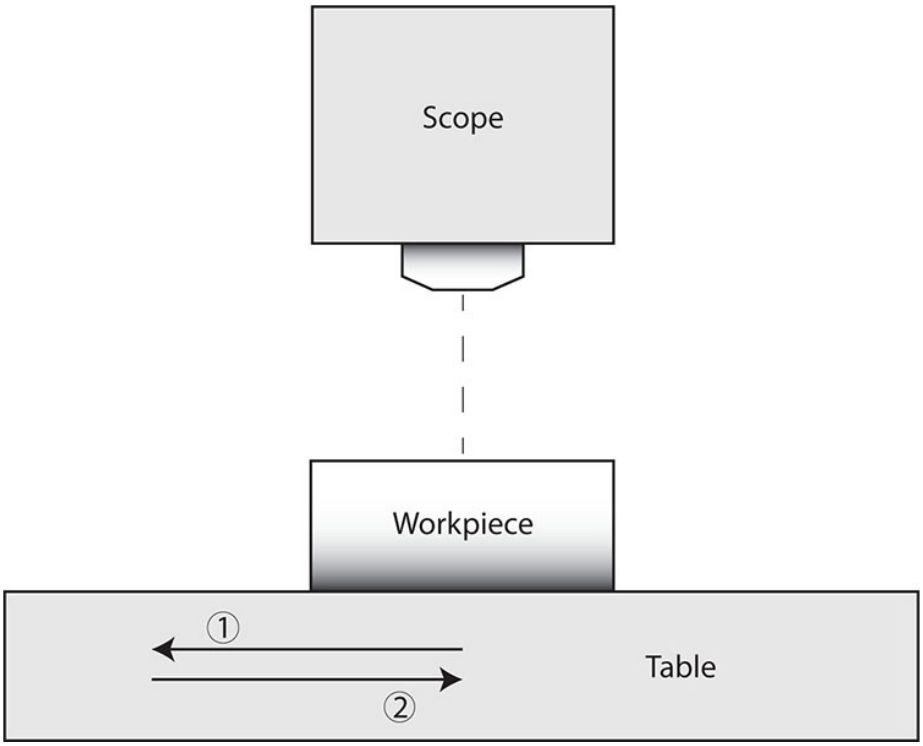


Fig. 2-18 Movements of a slide to eliminate the effects of backlash.

One feature of most optical scopes is that the direction of movement of an axis as seen through the scope appears to be reversed, compared to the actual movement of that slide. This is not a disadvantage, but simply a consequence of the arrangement of optical components in the scope.

How accurate is the position found using an optical scope? That depends on the magnification provided by the scope and on the thickness of the cross hairs, as well as on the size of the smallest jogging movement allowed by the CNC software. The scope provides adjustment to allow the cross hairs to be accurately aligned with the axis of the spindle so that even if the spindle is rotated a little, the cross hairs will remain aligned with the spindle axis. It also means that the cross hairs can be aligned with horizontal edges simply by rotating the scope or spindle, without losing the alignment of the central point. A good optical scope is likely to provide more accurate positioning than the average wiggler, but whether that increased accuracy is required is another matter.

Using a Laser Edge Finder

A laser edge and centre finder (Fig. 2-19) projects a beam from a laser held in the spindle on to the work and, in the basic model, this allows a ‘dot’ of light to be aligned with an edge or a mark on the workpiece. In other models, the beam produces a different pattern, such as set of crossed lines or concentric circles. The image can be seen even in quite bright light, making it an easy-to-use tool.

The critical factor for accuracy is the size and clarity of the image on the workpiece, and currently available laser edge finders use lenses and polarizing filters to project a small, sharply defined dot or a narrow line pattern that allows considerable accuracy and repeatability. Figs 2-20 and 2-21 show the difference between a basic projected dot and a dot projected through a polarizing lens.

When aligning with an edge, the technique is to split the dot on the edge by jogging the axes in small increments. As soon as the dot is on the edge and sheds light down the vertical face, it is right above

the edge and that axis can be touched off. Other patterns suit other applications; for example, a set of concentric circles is ideally suited for locating the centres of existing holes. The various patterns make the laser edge finder a versatile device for locating features and touching off. Figs 2-22 and 2-23 show the cross hairs in use to locate a position on a workpiece and to locate two adjacent edges.

Fig. 2-24 shows the concentric circle pattern (but note that this was photographed from an angle; the circles appear circular on the flat face of a workpiece if the beam is projected on to the workpiece at right angles).

Figs 2-25, 2-26 and 2-27 show the effects of changing the distance from the laser to the work and of focusing the beam, and Fig. 2-28 shows how the circular beam pattern can be used to align the axis with an existing hole, even where the edge of the hole has been chamfered. All in all, a versatile tool.

How accurate is the positioning achieved using a laser edge finder? That depends on how accurately the eye can judge the coincidence of the dot with a mark or an edge, or how accurately the circular displays can be aligned with existing holes. Like the optical scope, the laser can be calibrated so that the axis of the beam coincides with the axis of the spindle, so this is, potentially, an accurate device.

Most CNC programs allow continuous jogging movements of the Controlled Point, or any of a range of stepped movements, so that the movement can be controlled quite precisely. The normal sequence with any edge finder would be to reduce the movements to small steps as the edge finder gets closer to the face until it indicates that it has touched the face. It is a good idea to repeat the process more than once, to verify the exact location of that point, before touching off the axis. This also gives some indication of the accuracy of the process.

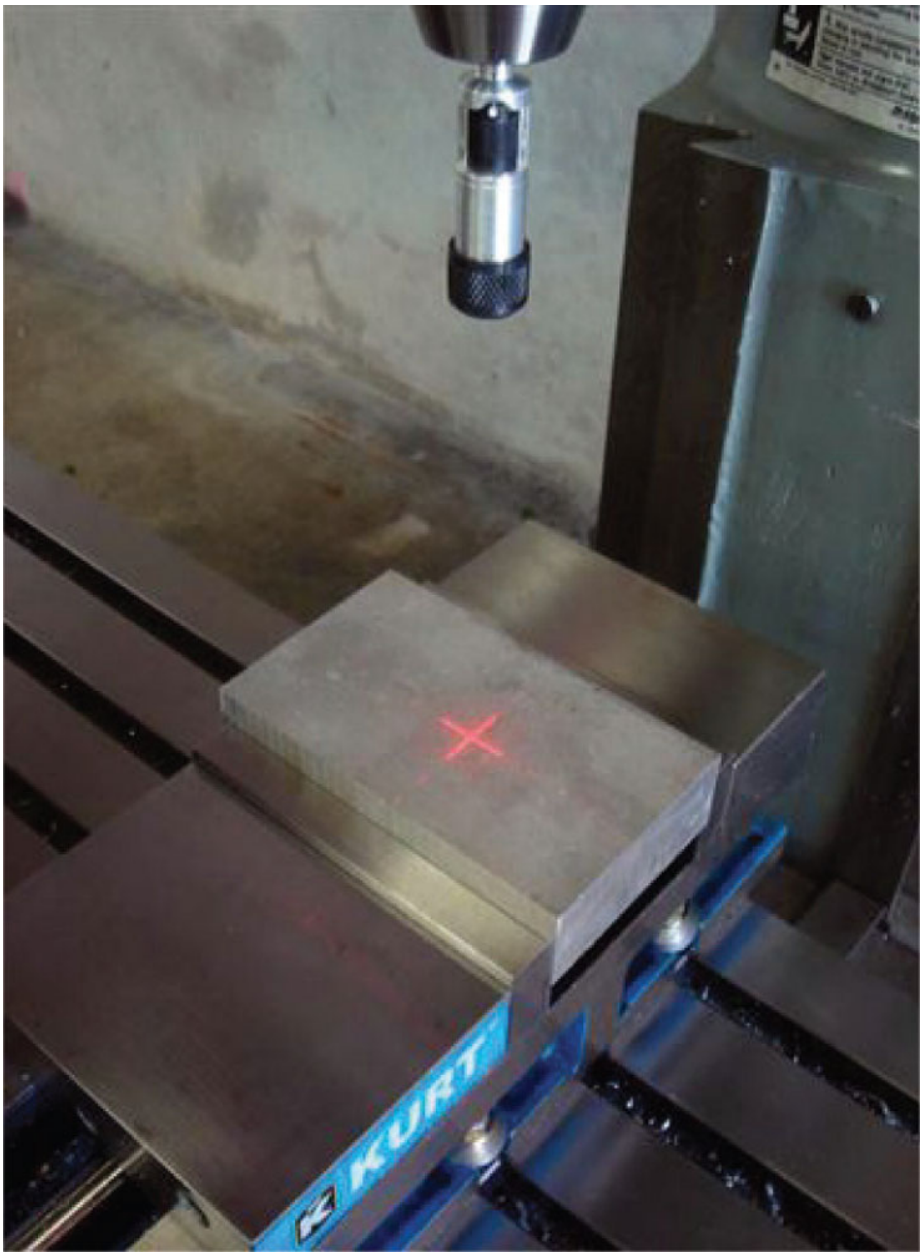


Fig. 2-19 A Laser Centre/Edge Finder projecting a cross-hair pattern on to a workpiece. (Photo: SDA Manufacturing)

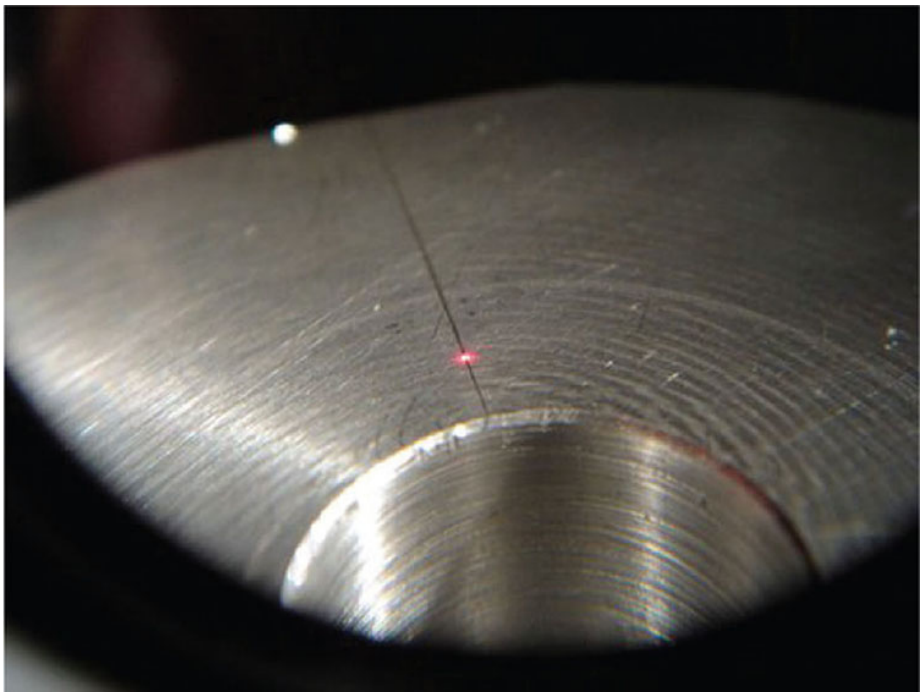


Fig. 2-20 A Laser Centre/Edge Finder without a polarizer, projecting a dot. (Photo: SDA Manufacturing)

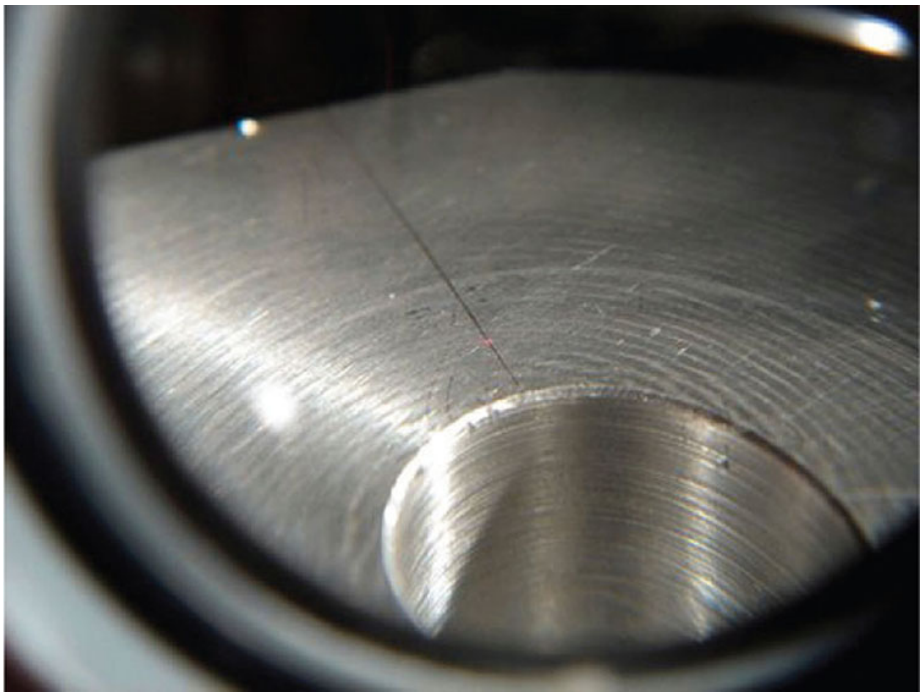


Fig. 2-21 A Laser Centre/Edge Finder fitted with a polarizer, projecting a dot. (Photo: SDA Manufacturing)

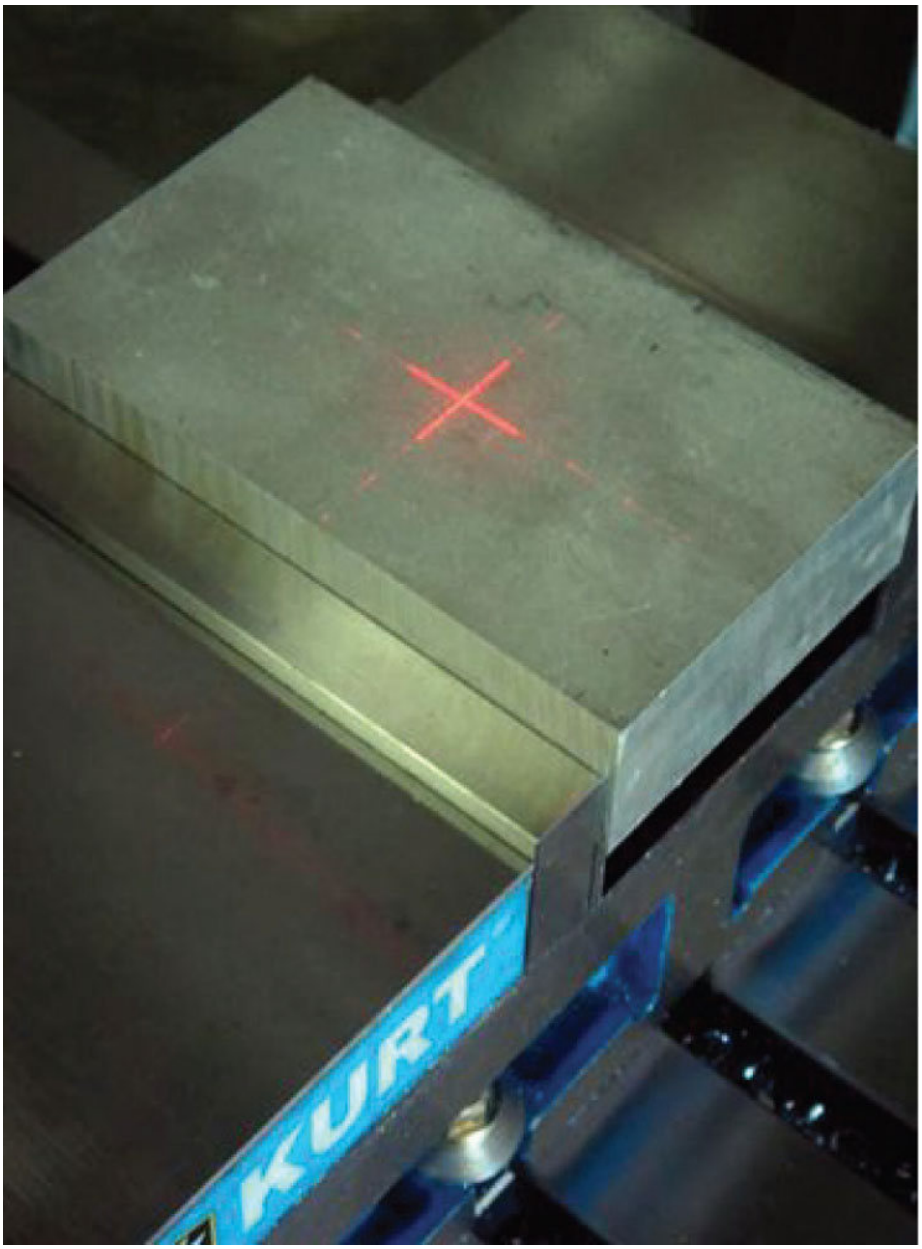


Fig. 2-22 Laser-generated cross hairs projected on to a workpiece. Note the length of the arms. (Photo: SDA Manufacturing)

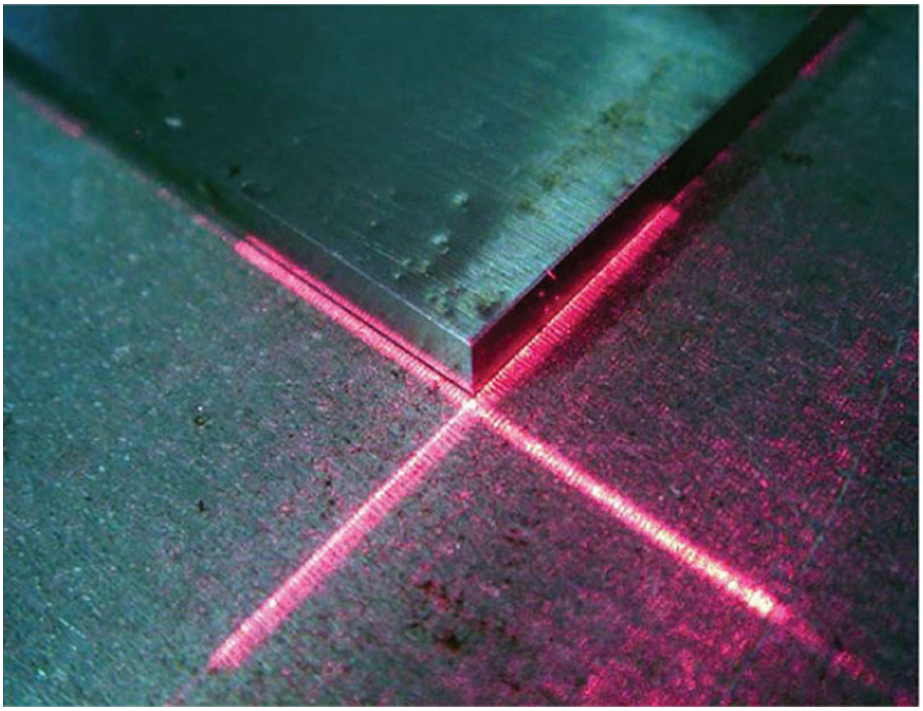


Fig. 2-23 A Laser Centre/Edge Finder with a cross-hair polarizing lens being used to align the corner and edges of a workpiece. (Photo: SDA Manufacturing)

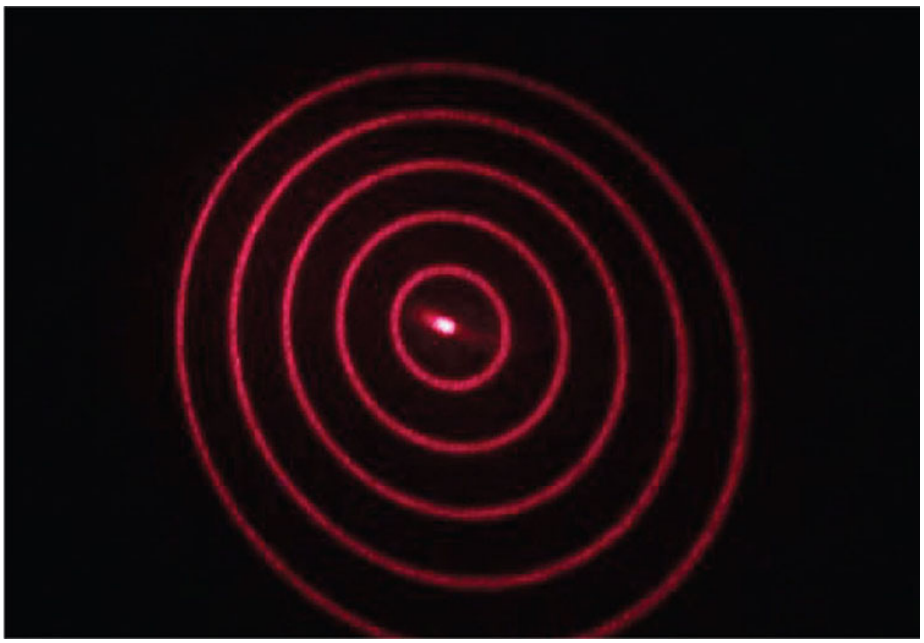


Fig. 2-24 A Laser Centre/Edge Finder fitted with a polarizing lens that projects a pattern of concentric

circles. The distortion is the result of photographing the pattern at an angle. (Photo: SDA Manufacturing)

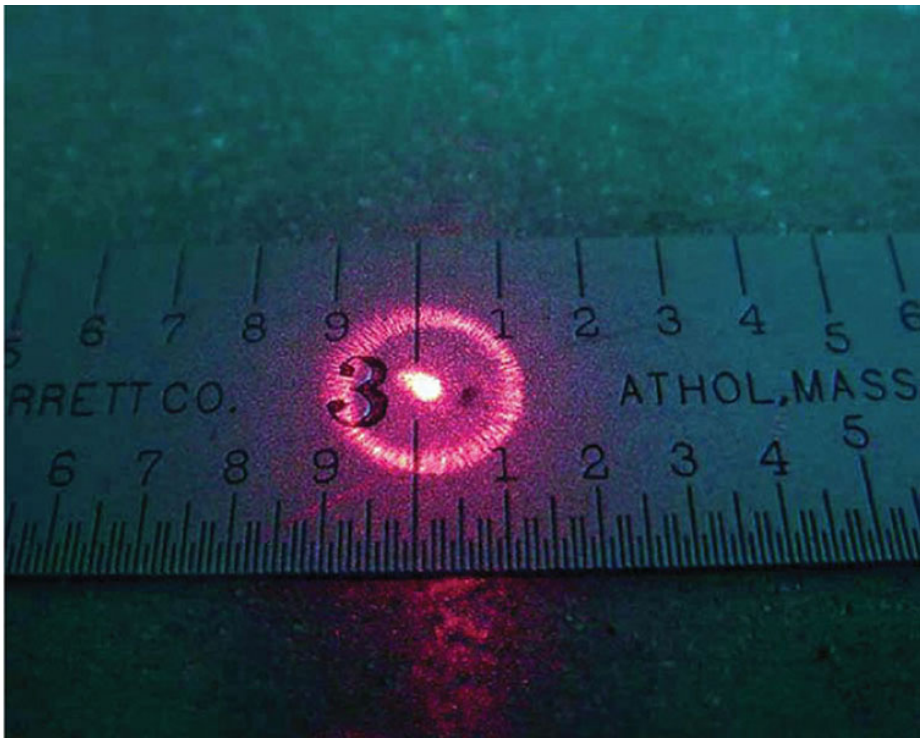


Fig. 2-25 If the laser is too close to the workpiece, the pattern is small and the beam is out of focus. (Photo: SDA Manufacturing)

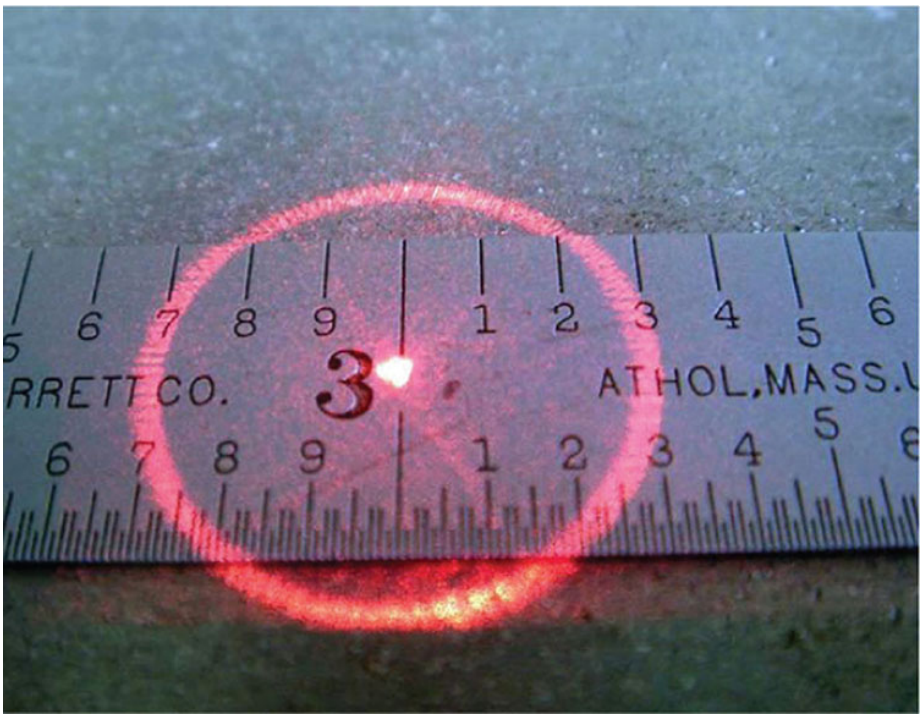


Fig. 2-26 Moving away from the work makes the pattern larger, but moving too far away puts it out of focus. (Photo: SDA Manufacturing)

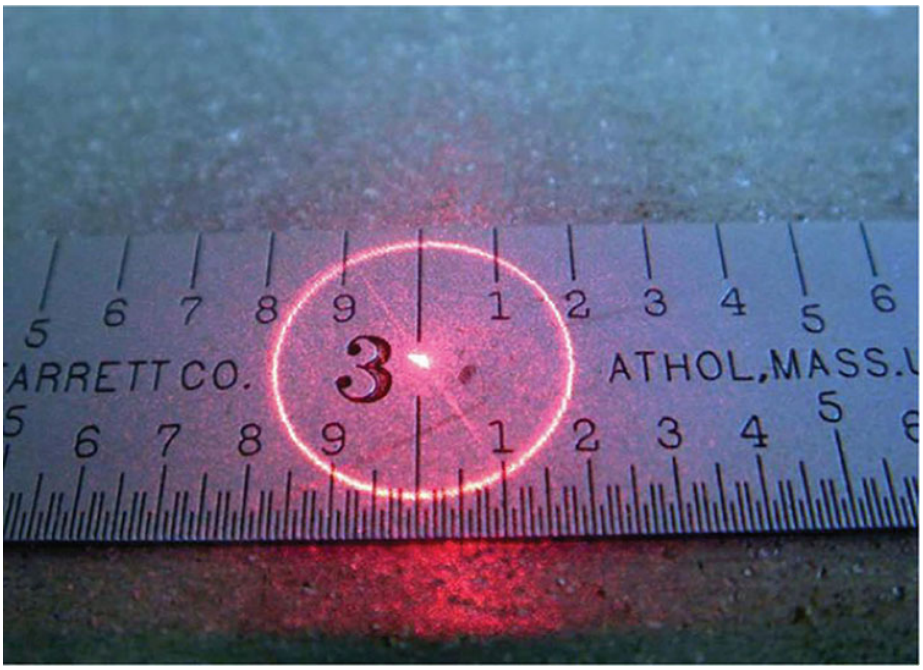


Fig. 2-27 Focusing the beam sharpens the pattern. (Photo: SDA Manufacturing)

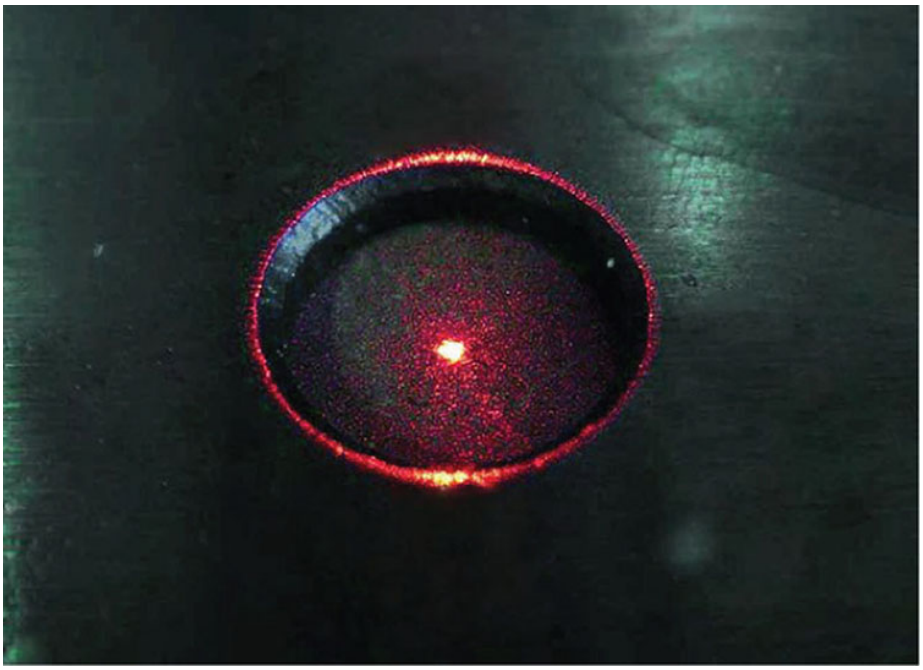


Fig. 2-28 Aligning a laser-generated circle with the edge of an existing hole. (Photo: SDA Manufacturing)

Touching Off the Z-Axis

The origin of the Z-axis is set using a range of different techniques. What is required is that the Controlled Point is located in relation to a suitable point on a workpiece. This is normally the top face or the top of the highest feature on the workpiece, but it might equally be the base of the workpiece.

The challenge is to set the end of the tool (that is, the Controlled Point) at that surface without damaging the tool or the work. Lowering the tool right on to the surface risks damaging the cutting edges, particularly if the end face of the tool carries cutting edges, like an end mill. Where the cutter is fragile, as in the case of a tiny carbide engraving tool with a tip width of perhaps 0.1mm (0.004in), lowering the tool directly on to the work will simply destroy the tool, so this is not a useful technique. Fig. 2-29 illustrates the use of a cylinder of known diameter (say 25mm or 1in).

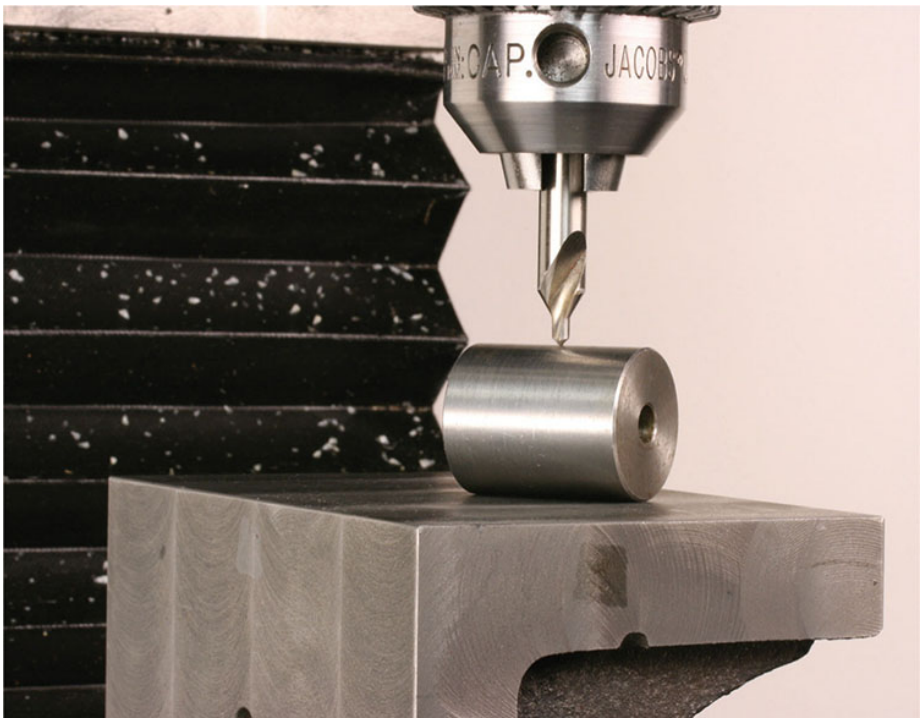


Fig. 2-29 Using a roller of known diameter as a gauge to set the height of a tool.

Lower the tool towards the work, stopping above the work at a distance of less than the diameter of the roller. Jog the tool upwards, testing whether the roller will just roll under the tip of the tool at each stage. It is quite easy to find the point at which the roller will just pass under the tool, while touching both the tool and the workpiece. The tip of the tool is now 25mm (1in) above the workpiece. The accuracy here depends on the jog increments you have used so, just as when touching off the X- and Y-axes, select smaller jog intervals as you approach the point at which the roller will just pass under the work. Touch off the Z-axis at this point, entering an offset of 25mm (1in). The offset is positive because the Controlled Point is 25mm above (that is, in the positive direction) the Z work origin.

Variations on this method include using a rectangular slip gauge and testing whether the gauge will just slip under the end of the tool as it is raised. Touch off the thickness of the gauge block.

If the surface where the origin is to be located is large enough, a Z height-setting gauge can be used, which may be electronic or

based around a dial indicator (Fig. 2-30) or may be an electronic version linked to the CNC software (Fig. 2-31) so that the height can be automatically entered into the software. The tool is lowered until it touches the probe (Fig. 2-32) and the axis is touched off, entering the height of the gauge (say 50mm or 2in).

Fig. 2-30 An analogue Z height gauge based on a dial indicator. (Photo: Allendale Electronics)

Fig. 2-31 An electronic Z height gauge. (Photo: Centroid Corporation)

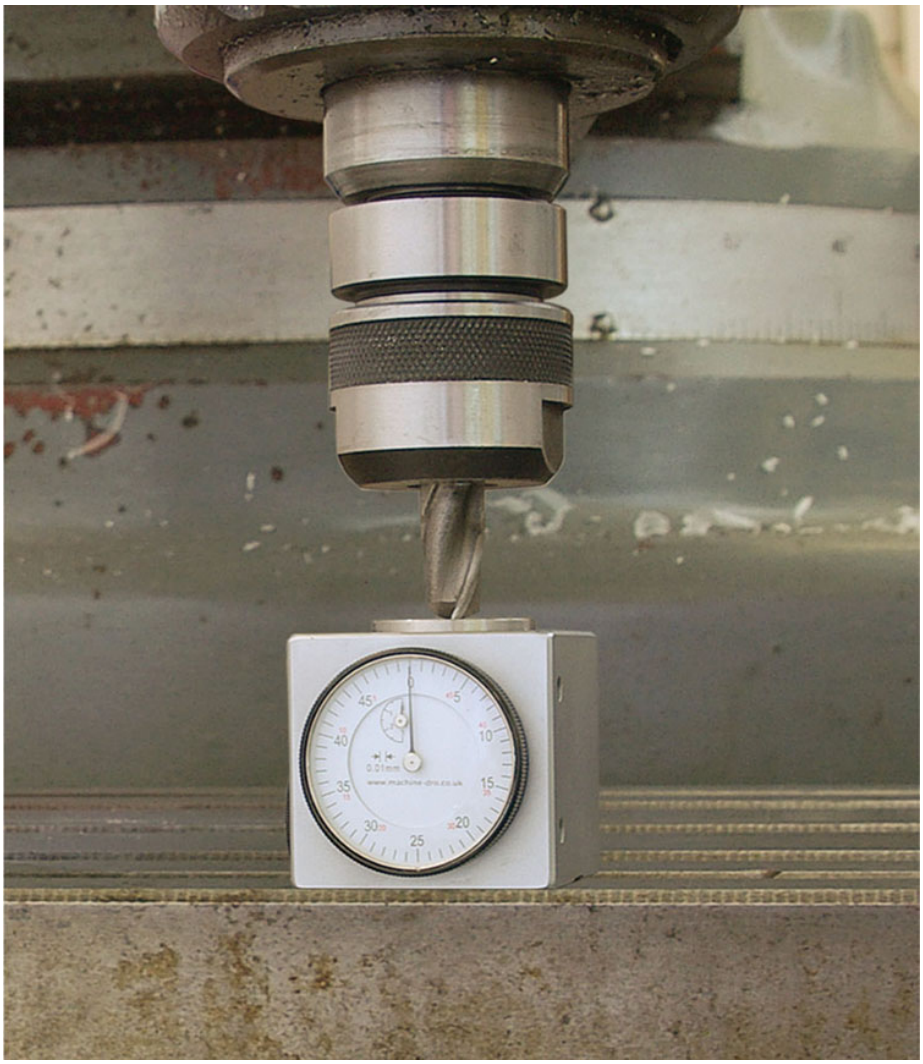


Fig. 2-32 An analogue Z height gauge in use. (Photo: Allendale Electronics)

If you take the time to establish the distance from the top of the inverted probe to the top of the vice jaws, you will then be able to enter that value when you touch off and the Z origin of the Controlled Point will then be at the top of the jaws, which is sometimes useful.

More on Offsets

There is more information on offsets later in this book, but for now it is enough to emphasize that CNC software can work with any one of

several coordinate systems, provided it is told which one to use. The machine works with a set of absolute coordinates whose origin is defined as Home, and all the other coordinate systems are based around that. Each new coordinate system is based on the absolute machine coordinates, but includes an offset for X, Y, Z and other axes. That means the origin of each coordinate system can be different and the offset is used to calculate that difference. Although there is a command to temporarily use machine coordinates, the user would not normally work with those absolute machine coordinates, but would always work in one of the offset coordinate systems. This is because there is usually a difference between the origin of the absolute machine coordinates and the position of the most convenient reference point on a workpiece. This is especially true where a machine does not use a home sequence with home switches.

The G code commands G54 to G59 are used to switch quickly to alternative coordinate systems, which is why the offsets are referred to as G54 offsets, G55 offsets, and so on. The G54 offsets (referred to as Fixture 1 offsets) are used to calculate coordinate system 1 and are the most commonly used alternative coordinate system, usually used as the basic work coordinate system.

Later, you will learn how to use multiple alternative coordinate systems to good effect when working with multiple workpieces.

A clock key embellished using a small end mill.

3 Basic Movement

In this chapter you will learn about:

- two ways of making your mill move.

This book assumes your CNC mill is set up and ready for work. Power up the mill, start the PC and run your CNC control software. In general, this book assumes you will not need finely detailed instructions for every step you must take to complete a task, but this initial section does give detailed instructions to get you started using Mach3 or LinuxCNC software because they are widely used packages. Most other packages work in a similar way, so although you may need to consult the manual for the software you are using, the functions will be the same; they may just be on a different part of the screen or in a different menu. It is what the software does that is important and not the particular buttons you need to press or menus you need to select. Once you have started, you should find that common sense and a bit of experience work wonders. The manual is useful too, of course.

To get started, the CNC software must be put into a state where it is prepared to drive the stepper motors or servos of your CNC mill. When the software starts initially, it goes into a ‘safe’ state and requires specific action from you to exit that state and begin driving the mill.

Program Run

Mach3: Clear the Reset by clicking on the big red button at the bottom left of the main screen (Fig. 3-1).

In the upper centre section of the main screen, click Ref All. If you have a home sequence defined, that should move the machine slides towards the home switches, trip the switches and set Home as (0, 0, 0). If you do not have a home sequence defined, or have chosen not to use home/limit switches, no movement will take place and you should click the buttons to Zero X, Zero Y, Zero Z and Zero 4.

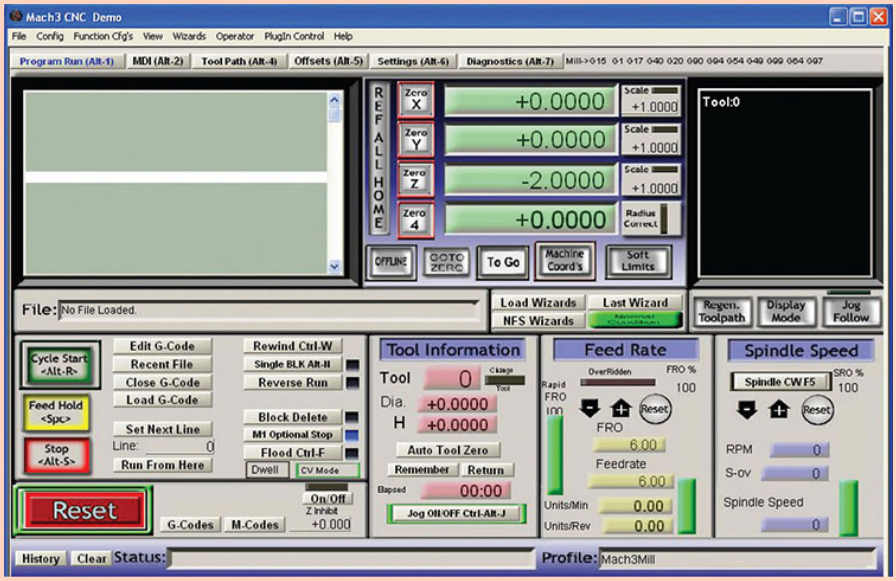


Fig. 3-1 The main Mach3 screen.

Manual Control

LinuxCNC: Clear the red E-Stop at the top left of the main screen (Fig. 3-2) by clicking it, and then do the same to Enable Machine Power (next button on the right).

If you have a home sequence defined, there will be a Home All button halfway down the screen. In that case, click Home

All to move the machine slides towards the home switches, trip the switches and set Home as (0, 0, 0).

If you do not have a home sequence defined, or have chosen not to use home/limit switches, the button will read Home Axis.

In that case, for each axis in turn:

- ***click its radio button (the little round circle);***
- ***click Home Axis.***

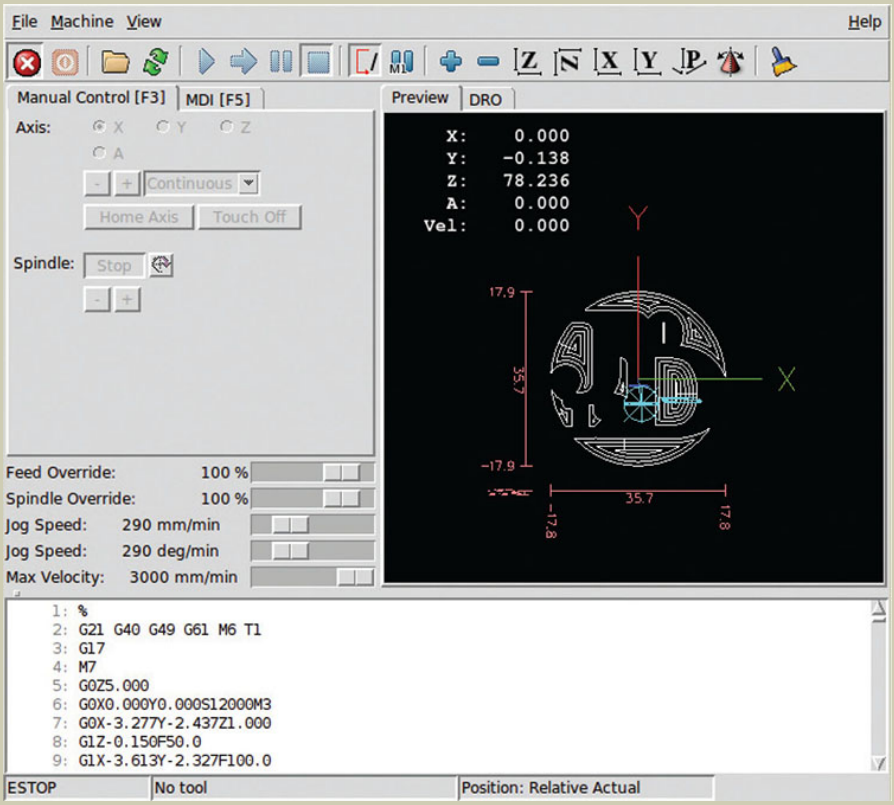


Fig. 3-2 The main LinuxCNC Axis screen.

Both programs have at least three ways of controlling the mill. The first is **program** mode, the second is **MDI (manual data input)** mode, and the third is the **jog** facility. You will learn more about the

different modes as we go along, but the important thing at the moment is to get the machine moving.

JOGGING

Jogging is performed by pressing keys on the keyboard to move the Controlled Point in various directions. Remember that when a CNC mill moves, it is the Controlled Point that is moving in relation to the work, so if the X and Y slides seem to move in the opposite direction to what you expect, try thinking about the direction in which the head is moving compared to a fixed point on the table.

Program Run

Mach3: At the main Program Run screen (accessible via the tabs at the top of the screen), press the TAB key ⇥ to reveal the jog controls (Fig. 3-3). This controller can be hidden by pressing TAB again. Leave it visible for now.

Clicking the Jog Mode button (halfway down the controller) will select either Cont (continuous) or Step. Set it to Cont.

In the Button Jog section, press the buttons and watch the slides move. As you jog, you should be able to see the DRO on the main screen and the slide position values shown there should change as you jog.

X and Y control the X and Y axes of the table.

Z controls the vertical movement of the head (or possibly the table, if you have a knee mill like a Bridgeport).

4 controls a fourth axis (such as a rotary table) if one is fitted.

Jogging can be more conveniently controlled using keys on the computer keyboard, and it is especially convenient if the keyboard has a numeric keypad (Fig. 3-4).

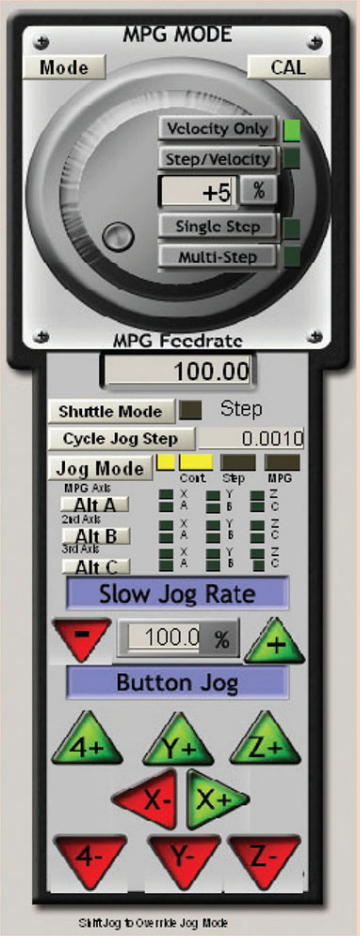


Fig. 3-3 The Mach3 pop-out Jog Controller.

Fig. 3-4 Jog keys on the numeric keypad. The 'A' rotation keys may vary or be undefined.

Press 4 or 6 to jog the X axis (these are sometimes labelled with left and right arrows), 8 or 2 to jog the Y axis (these are sometimes labelled with forward and back arrows), and 9 or 3 to jog the Z axis (these are sometimes labelled PgUp and PgDn).

If you have a fourth axis fitted, the keys that control this axis will have been chosen using the ConFig. > System Hotkeys menu. Check there to see if they have been set and which ones have been chosen. The keys are identified by their ASCII codes.

Manual Control

LinuxCNC: At the main screen (the Manual Control tab under the menu bar), the word or number just above the Home Axis (or Home All) and Touch Off buttons is part of a menu that allows you to select the jog increment. Use the black triangle to set it to read Continuous.

Select an axis by clicking in its radio button, then press the − and + buttons to jog that axis. This is not terribly handy, but you should check it works.

Jogging can be more conveniently controlled using keys on the computer keyboard and it is especially convenient if the keyboard has a numeric keypad (Fig. 3-4).

Press 4 or 6 to jog the X axis (these are sometimes labelled with left and right arrows), 8 or 2 to jog the Y axis (these are sometimes labelled with forward and back arrows), and 9 or 3 to jog the Z axis (these are sometimes labelled PgUp and PgDn).

If you have a fourth axis fitted, the [and] keys on the main keyboard can be used to jog that fourth axis.

Continuous jogging is frequently used to control the initial overall position of the Controlled Point in relation to the slides. Jogging in increments or steps is used to control the position more accurately, so you can imagine moving the slides large distances quickly, then moving in small precise steps towards a final position. Select Step mode and a small step size, and jog.

Program Run

Mach3: On the pop-out controller, click the Jog Mode button to select Step mode, then click on the Cycle Jog Step value and enter a step size (0.1 for example).

The Jog Speed can be set in the Slow Jog section using the arrow buttons on the controller. The value here sets the jogging speed to a percentage of the Rapid Traverse Rate, which is set in the Feed Rate section of the main screen.

- ***TAB to put the controller away.***
- ***Type a value into the Feedrate box.***
- ***TAB to bring the controller back into view.***

Manual Control

LinuxCNC: Change the jog increment from Continuous to a step size, by selecting one of the other values using the drop-down menu (0.1 for example).

The Jog Speed can be set using the Jog Speed slider further down the screen.

Use the jog buttons or keyboard keys to jog the slides. They should move in small discrete steps, 0.1 units at a time (0.1mm, if the software is set up to use millimetres, or 0.1in, if it is set to use imperial units).

MDI MODE

The MDI screen allows the slides to be moved under direct typed commands, so it is a stepping stone to running a program. Select the MDI tab to enter MDI mode. On the screen you will see a long, thin input window.

To move a slide, we will use G code commands. These are explained in more detail in the section on linear programming (Chapter 5), but it is easy enough to use some of the simple G code commands for movement now.

Before doing that, jog the mill so that the table sits more-or-less centrally within the limits of its X and Y travel. Do the same for the Z axis, so that the head sits well off the table but well clear of the end of its upper limit of travel. It would be convenient if this was set to be the origin for an imaginary workpiece at (0, 0, 0), because that would give us an easy-to-use reference point as we make some simple movements.

To set the work origin, go back to the main screen Program Run or Manual Control.

Program Run

Mach3: In the upper centre section of the screen, click on the Zero X button, the Zero Y button, the Zero Z button and the Zero 4 button. This should set the read-out to show 0 for each axis.

Manual Control

LinuxCNC: For each axis in turn:

- ***click on its radio button;***
- ***click the Touch Off button, enter 0 and click OK.***

In the MDI window, enter a feed rate by typing F100 then press the Return key ↵. In general, press Return after each command or line of instructions to have it carried out.

The G0 command (that is the digit zero) is used to move a slide at a rapid feed rate. The command needs to identify the axis of movement and the target coordinate value for the position.

The X, Y and Z values on this page are in millimetres. If you are working in inches, divide them by 25. Before moving any slide, check

the available travel, especially when moving the head (Z axis).

THE ESC KEY

The ESC key ESC at the top left of the keyboard can be used to stop the command currently being carried out. It is sometimes useful to just hover over the ESC key if you are not sure of the motion of a slide.

Type G0 X100 to make the Controlled Point move along the X axis to coordinate 100. Bring it back by typing G0 X0.

Try G0 Y50 then G0 Y-50 and G0 Y0. Your software is quite capable of moving two axes at once, using coordinated movement. This means each slide will move at a rate that will cause it to reach the end of its movement as the other slide reaches the end of its movement. This means it can cut at an angle in the XY, XZ or YZ planes.

G0 X100 Y50
G0 X-100 Y-50
G0 Z-25
G0 Z25

The G1 command is used to move at cutting feed rate.

In fact, the mill can manage to carry out coordinated movement in three axes simultaneously.

G1 X0 Y0 Z0
G1 X-25 Y25 Z25

This means it can machine at a compound angle in three-dimensional space.

SPINDLE CONTROL

On some mills, the spindle motor is under software control; others rely on manual control of that motor. If your mill is under software control, you can experiment with spindle control now.

Program Run

Mach3: Go to the Program Run screen. In the Spindle Speed section, use the + and − arrows on the screen to increase and decrease spindle speed.

The spindle speed can also be set by typing a value directly into the S value space.

The direction of rotation can be changed by clicking the Spindle CW (or CCW) button.

Manual Control

LinuxCNC: At the Manual Control screen, the spindle direction can be chosen from the menu next to the Spindle button, and the spindle can be started using the Spindle Start button. The + and − buttons increase or decrease the speed. Depending on how your machine is set up, there may be a spindle brake button.

In MDI mode, the s command can be used to set the spindle speed, so S1000 would set the spindle speed to 1,000rpm. This command only sets the speed; it does not make the spindle rotate.

M3 starts the spindle rotating clockwise and it will accelerate up to the operating speed set by the S command.

M4 starts the spindle rotating anticlockwise and it will accelerate up to the operating speed set by the S command.

M5 stops the spindle.

Try these commands now, in MDI mode.

Your mill may indicate the spindle speed on the DRO display, but only if it has a sensor and pulse generator fitted.

Using the G0 commands, typed in MDI mode, it would be possible to use the mill as an excellent coordinate drilling machine. At each position where movement stops, you could use the handle feed on the head to drill a hole, or a G1 command to move the Controlled Point downwards. Use a slow speed, such as F50 or less, until you are comfortable with this. Practise drilling into soft material at first. Wood or plastic are ideal.

Project 3.1

Writing Movement

The accurate control of movement is an essential part of the machining process, and guiding the Controlled Point manually using MDI mode is a good way to start because it allows pauses between movements to give the operator time to think. It is a good way to experiment gently.

TRACING A PATH

Assuming you have homed your machine, put a sheet of material on to the mill table, clamping it so that it cannot move. Tape a sheet of paper to the usable area.

If you can find a pen with a fine, soft tip and a cylindrical body, or a short pencil, put a drill chuck in the spindle and grip the pen or pencil in the chuck (Fig. 3-5). If you are using a pencil, put a piece of card under the paper; otherwise the pencil point will blunt quite rapidly.

Jog the pen down until it just touches the paper. Touch off Z at that point. Then retract the pen by jogging Z upwards or by typing a G0 command in MDI mode.

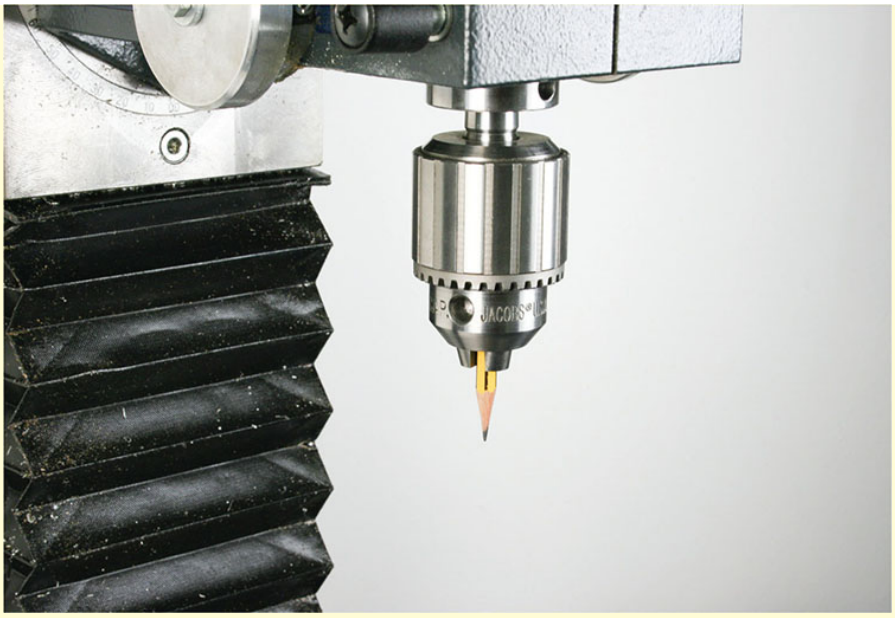


Fig. 3-5 Use a pencil or pen in a drill chuck to draw the toolpath.

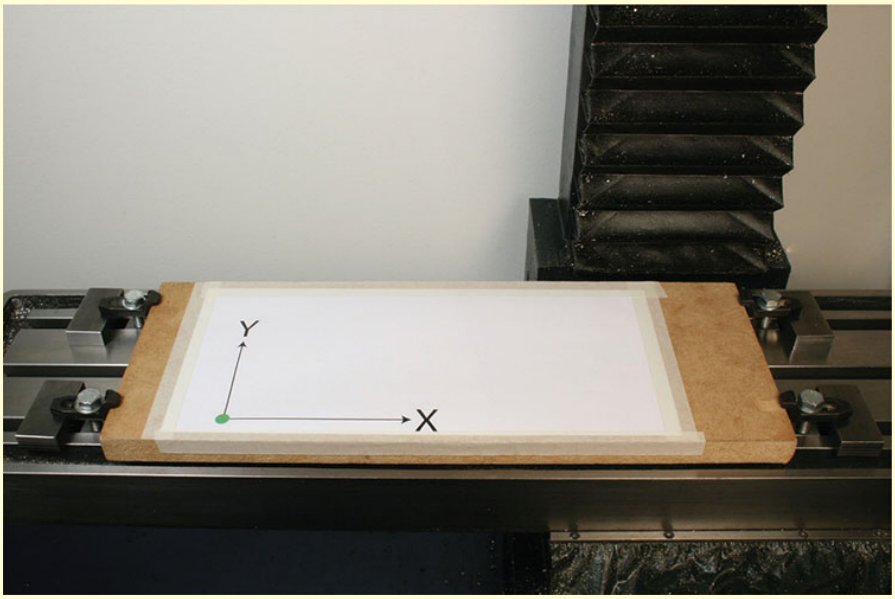


Fig. 3-6 Set the origin of the paper to the bottom left corner.

Notice the Z reading where the pen is clear of the clamps on the table. That is the ‘safe Z’ position and it can be made a convenient round number.

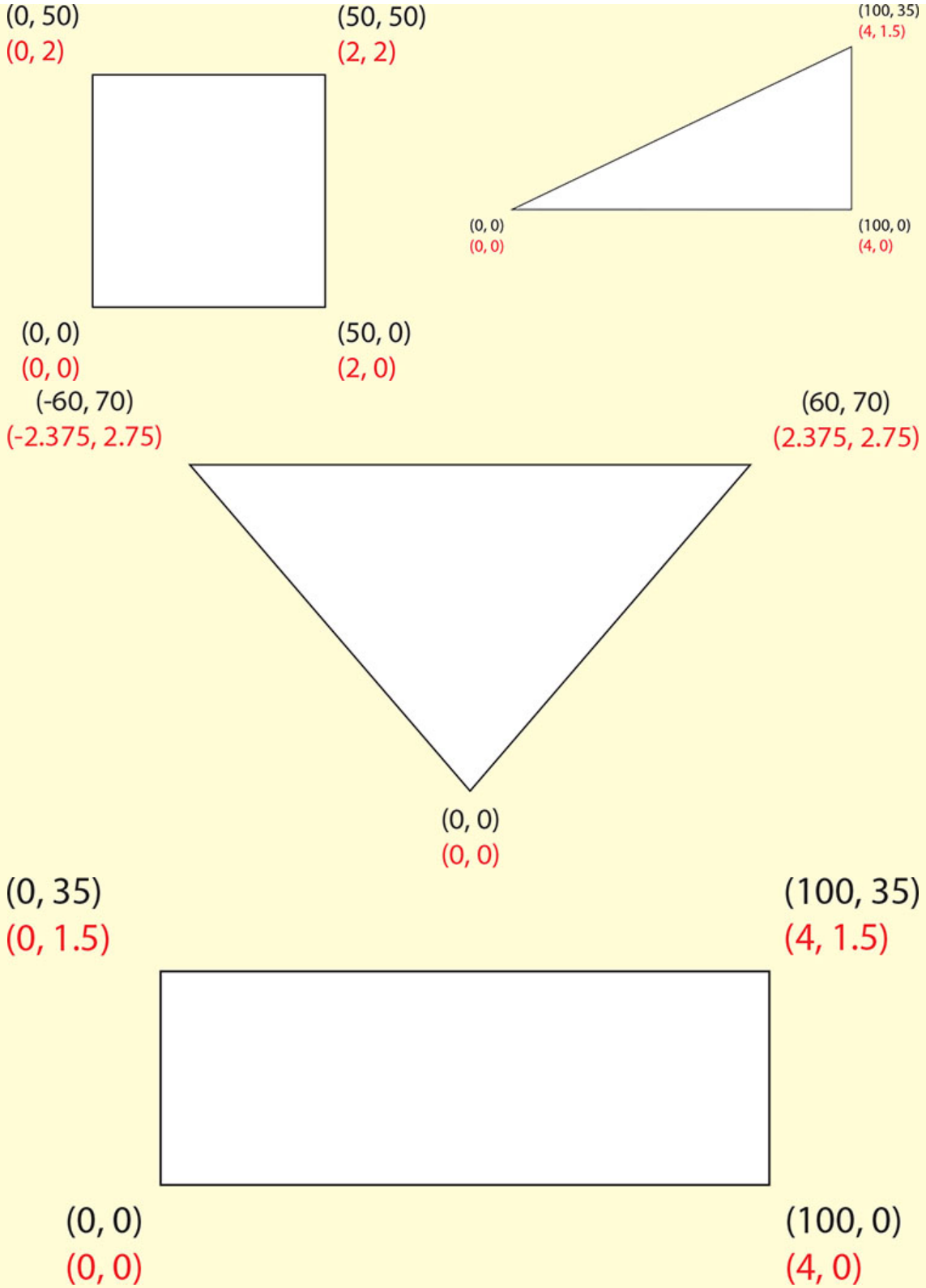
Bring the pen close to, but not touching, the paper, then jog to position the pen above the front left corner of the paper, as near as you can judge by eye. Touch off the X and Y axes, setting the origin at a convenient point near the front left corner of the usable area of the sheet (that is the area clear of clamps), as shown in Fig. 3-6.

Using commands in MDI mode, try drawing the shapes shown in Figs 3-7 to 3-10.

Take the paper off the mill; sign it and hang it proudly on the workshop wall.

Put a new piece of paper on the usable area, and tape it down.

The next set of shapes (Figs 3-11 to 3-14) are symmetrical and might benefit from having the origin in the positions indicated by the green circles. This makes it easier to work out the coordinates of the points the Controlled Point must visit.



Figs 3-7 to 3-10 Shapes to create using a pencil or pen held in the drill chuck. Dimensions shown in black are millimetres; dimensions

shown in red are inches.

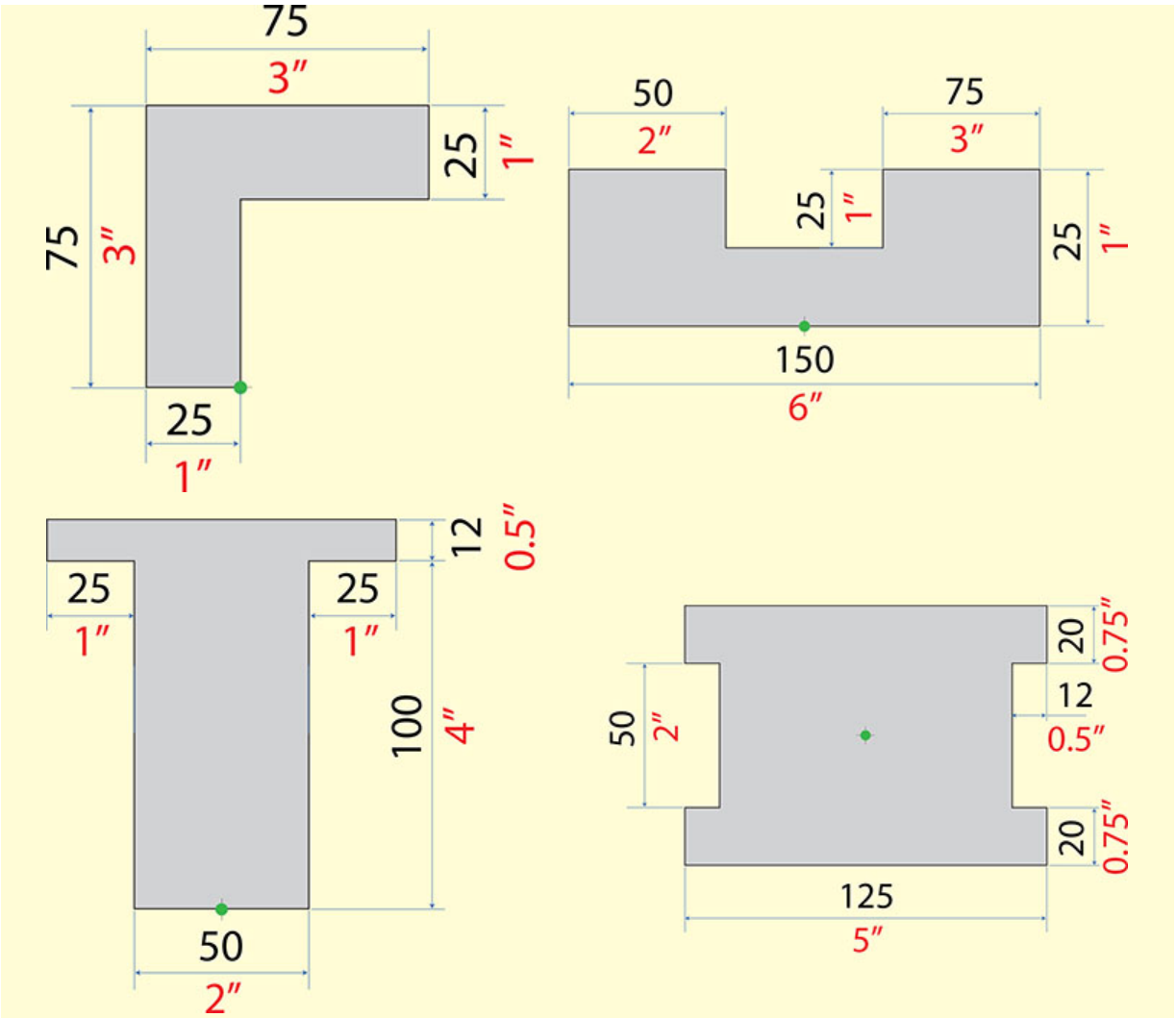
For each shape, jog the Controlled Point to a suitable position on the sheet, and touch off X and Y at that position to make that the origin (0, 0). The Z axis will not need to be touched off unless you have used a much thicker piece of paper.

Then draw the shape. Although this method might seem a little slow and quite simple, it can be useful, occasionally, if you are in doubt about the path of the Controlled Point.

However, do remember that the drawn line has no significant thickness (unless you tried to press the pen a bit too far downwards…). If there was a cutter of any size in the chuck and the Controlled Point followed the same path, the edges of the shape being machined would be offset to one side by half the diameter of the cutter. There is more on that in later chapters.

All those drawings can go in your portfolio; they form part of your emerging collection, and might form part of an exhibition of work in the workshop. Or not.

More interestingly, can you make the pen sign your work? There is an example on www.cncintheworkshop.com. That site also has plans for a spring-loaded pen or pencil holder that you can make if you have a lathe.



Figs 3-11 to 3-14 Shapes to create using a pencil or pen held in the drill chuck. Dimensions shown in black are millimetres; dimensions shown in red are inches. The green circle indicates the origin.

Project 3.2

Small Multitool Holder

The workbench can easily become cluttered with all those essential tools and accessories, so it is a good idea to have a

place for everything and to keep everything in its place; in an ideal world at least.

A multitool can be very useful, but the little rotary tools it uses can proliferate and sometimes seem to get everywhere. Drills, polishing mops, grinding discs and little saws are very handy, but need to be organized (Fig. 3-15). One common shaft size for these tools is either 3mm or ⅛ in (3.2mm), so a simple wooden or plastic block with some drilled holes can help organize these items (Fig. 3-16).

Fig. 3-15 Rotary tools of all shapes and sizes need to be organized.

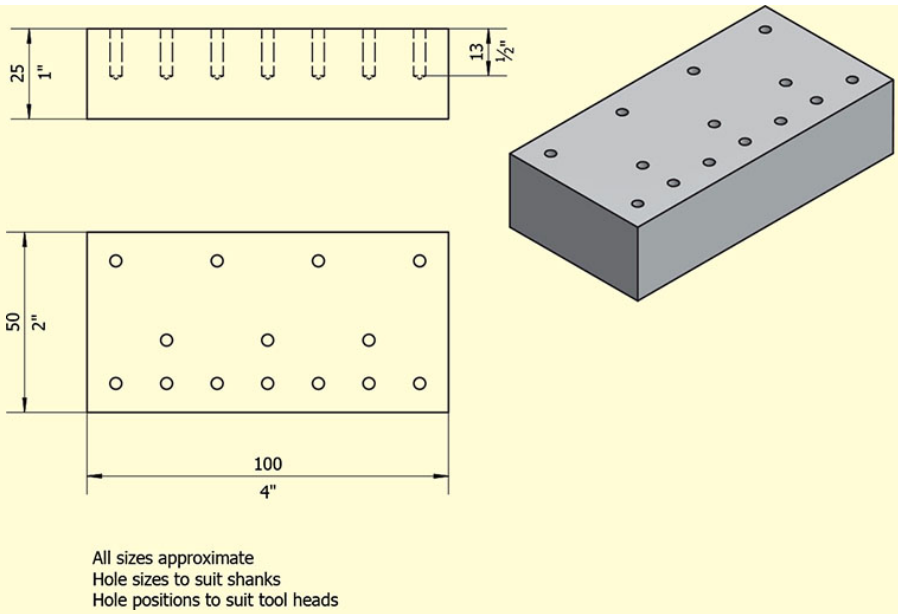


Fig. 3-16 A rotary tool holder.

Material

Wood is fine, but plastic or even aluminium would work well.

The drawing specifies a 50 × 100mm (2 × 4in) block, 25mm (1in) deep, but you can make the block to suit the size of material you have. You can make the holes any suitable size to suit your own tools, to accommodate smaller or larger shafts, or individual drill sizes.

Tools

- Twist drill to suit the size of hole. Allow a little clearance so that the accessories will not be gripped too tightly in the holes.
- For a 3mm shank, use either a 3.1mm or a ⅛ in drill.
- For a ⅛ in shank, use either a 3.3mm or a No. 30 drill. A 9/64 in drill is perhaps a little too big.
- Optional: a small centre drill. For 3mm (⅛ in) tool shanks, use a BS3; for the smaller shanks, use a BS2. The idea is simply to use a centre drill that has a pilot diameter of slightly less than the tool shank diameter.

Speeds

Run the drill at up to 4,000rpm. Slower speeds will work fine, because you will be feeding the drill into the work manually and can feel for the feed rate to suit the material and the speed.

Method

Fig. 3-17 shows how you might clamp the block to the mill table, but you must try to position the clamps so that they do not cover any of the hole positions. The block can be set approximately parallel to the X axis by sighting over the back edge at an angle, to allow you to compare the back edge of the work with one of the T-slot edges.

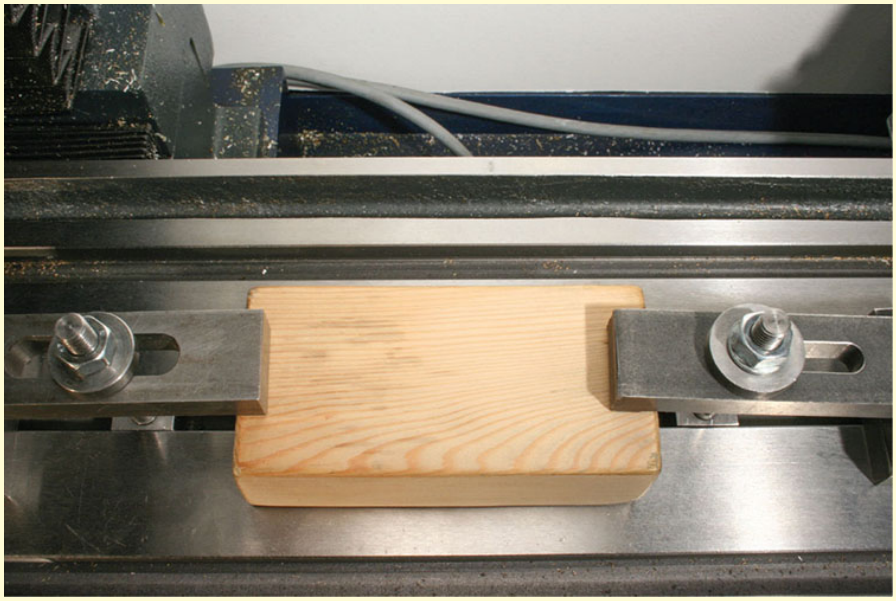


Fig. 3-17 Align the block by sighting over the back edge to a T-slot.

Fig. 3-18 shows the pattern of holes in the top face and Table 3-1 gives some suggested locations for the holes. Measurements are from the front left corner, with row 1 towards the front and row 3 towards the back.

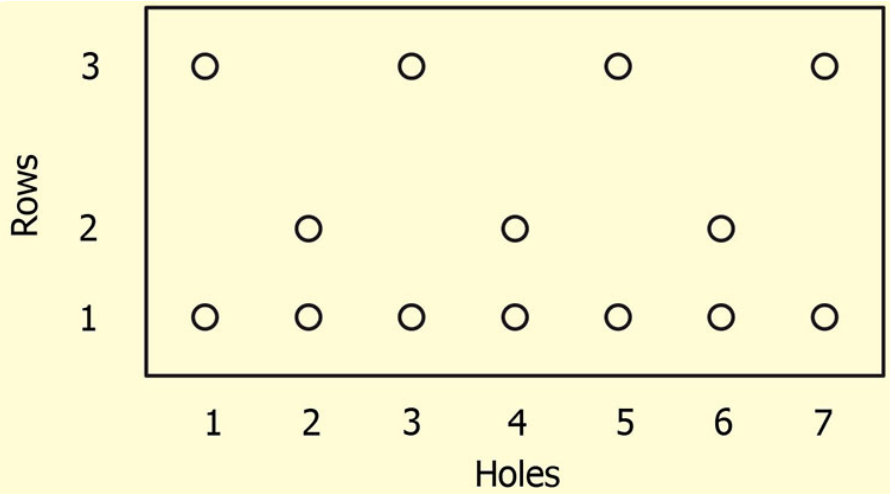


Fig. 3-18 Reference numbers for holes on a tool-holder block.

Table 3-1

Row-Hole	Coordinates (mm)		Coordinates (in)	
number	**X**	**Y**	**X**	**Y**
1-1	8	8	0.5	0.3
1-2	22	8	1	0.3
1-3	36	8	1.5	0.3
1-4	50	8	2	0.3
1-5	64	8	2.5	0.3
1-6	78	8	3	0.3
1-7	92	8	3.5	0.3
2-1	22	20	1	0.75
2-2	50	20	2	0.75
2-3	78	20	3	0.75
3-1	8	42	0.5	1.7
3-2	36	42	1.5	1.7
3-3	64	42	2.5	1.7
3-4	92	42	3.5	1.7

When you are moving from one hole to another, remember the Controlled Point will move in a straight line, so think carefully

about whether the tool will hit a clamp. It is a good idea to retract the tool above the height of the clamps, before moving to the next hole. That retracted height above all obstructions is called the SAFE Z height.

Going further

Fig. 3-19 shows a more extended version of the basic idea. Make the layers out of slightly thinner material so that the heads of the brushes do not foul the next higher block or the shanks of the tools in its first row. Drill the holes in each layer of the material before gluing it together. If the material is too wide (back to front) for your mill, try setting the work origin at the top right and let the extra material stick out over the front of the mill table, provided this is a clear space and the material will not hit anything as the table moves. You will need to use negative values for each of the coordinates in the table; so hole 1-1 will have coordinates (−8, −8) and the appropriate command would be G0 X-8 Y-8. There is an example of working this way in Chapter 7.

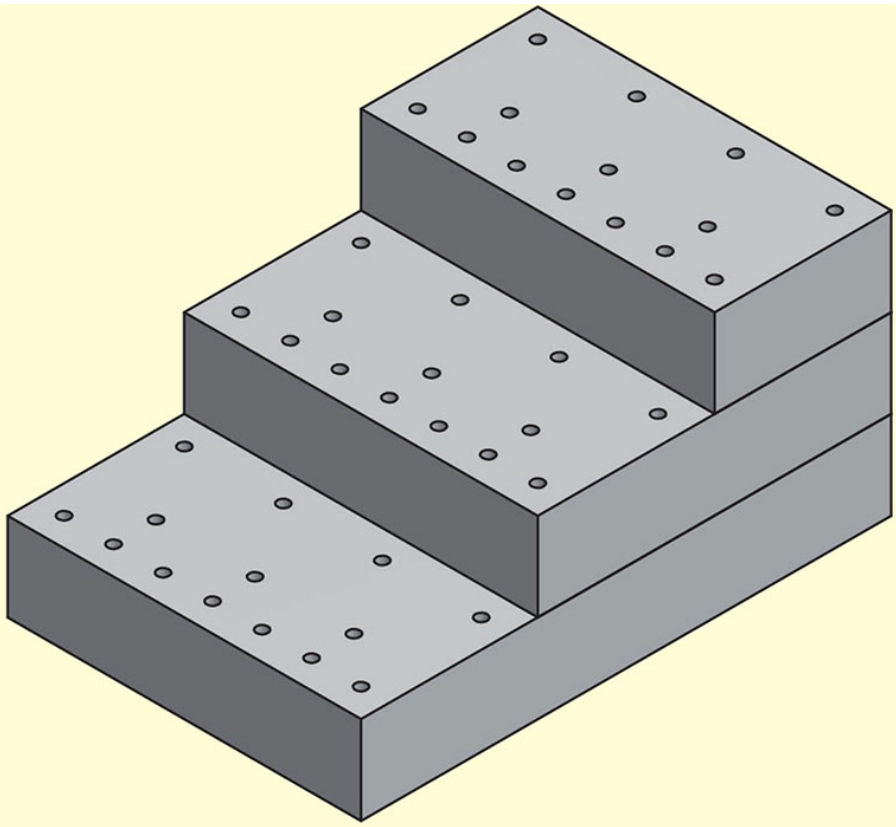


Fig. 3-19 A multi-level holder.

A selection of tooling for a Tormach mill. (Photo: Tormach LLC)

4 Tooling

In this chapter you will learn about:

- the kinds of tools commonly used in a CNC mill;
- how to work out the best speeds and feeds to use.

Most tooling for the mill is designed to be held in some kind of holder in the head of the mill and to cut as the spindle rotates. There are some exceptions, but that is the basic method.

HOLDING TOOLS

Tool holders usually locate inside the rotating spindle of the mill head. The spindle normally has an internal taper so that the tool holder can locate accurately and safely, time after time. Typical tapers are Morse tapers No. 1, 2 or 3, and R8 (the same as Bridgeport mills), although larger machines may use an ISO taper. The size of taper often reflects the size of the spindle and the spindle bearings, so that on a small machine there may only be room for the Morse No. 1 taper, while larger machines have larger spindles and spindle bearings and can accommodate a larger taper like MT3 or R8.

Most mills will have a hole right through the length of the spindle to allow a drawbar to be fitted. This is a security device that ensures that any tool holders fitted to the spindle taper cannot work loose due to vibration.

Drills can be held in a drill chuck (Fig. 4-1) in the same way as they would be in a drilling machine. These chucks are designed to

hold drills, reamers, and taps, but are not suitable for any other type of cutter.

Fig. 4-1 A drill chuck.

Like most tool-holding devices, drill chucks have a body that has jaws to hold tools and a shaft that locates in the spindle taper. The shaft is usually a permanent fit in the body of the chuck. Chucks are sometimes supplied with shafts already fitted, but bare chuck bodies normally have a short taper in the rear (often a Jacobs taper) so that a shaft can be chosen to suit the chuck taper and the mill taper, and the chuck is then permanently assembled on to the shaft. Drill chucks that screw on to a shaft are usually not sufficiently accurate for use in a milling machine.

The shaft normally has a threaded hole down its length and this will be tapped for a drawbar. These are usually standard thread sizes.

Once the chuck and shaft have been inserted into the mill spindle, a drawbar is screwed into the shaft, from the top, to secure the shaft and chuck. The tool is inserted into the chuck and the chuck key is used to close the jaws firmly.

A handy variation is the keyless chuck, which can be tightened by hand by rotating a collar on the chuck. A good-quality keyless chuck is a useful item, but as with any other tool holder, a poor-quality version is a liability.

STANDARD DRAWBAR THREADS

The most common drawbar threads depend on the taper of the spindle as shown in Table 4-1.

Table 4-1

Taper	**Thread**
R8	7/16 – 20 UNF
Morse 2	M10 or 3/8 × 20TPI Whitworth
Morse 3	M12 or 3/8 × 20TPI Whitworth

What Constitutes Quality in a Tool Holder?

First, it must hold the tool securely, even when the tool is vibrating or tending to be pulled into the work. Second, it must hold the tool so that its axis coincides with the axis of the spindle.

Good-quality tool holders do both, but a poor-quality tool holder usually fails to satisfy this second requirement. The result is that the tool cuts a larger hole or takes a broader cut than it should, and the cutting action is impaired because only part of the tool is cutting effectively.

Milling cutter holders can be as simple as the one shown in Fig. 4-2. This design of holder is simply a taper ending in a head that has a hole to suit the shaft of the cutter, and a grubscrew to clamp the shaft in the hole. Many small milling cutters have flats on their shafts

for this purpose, so although the design of this kind of tool holder is quite simple, it can be an effective and relatively low-cost method of holding tools.

Fig. 4-2 A simple milling cutter holder.

The relatively low cost means that instead of having one holder and changing the cutters, the most frequently used cutters can stay in their own holders. This has important consequences for tool setting and tool changing, as we will see later.

There are two commonly used types of tool-holding collets in common use. The first is a purpose-designed milling collet system (Fig. 4-3) consisting of a shaft and body, a removable nose piece and individual collets for the common size of cutter shafts. This collet system is modelled on the Clarkson Autolock chuck and is designed to hold milling cutters that have a thread on the end of their shaft (Fig. 4-4) and a locating dimple in the end. Cutters use one of a small range of shaft sizes, but all sizes have a Whitworth form thread of 20tpi (threads per inch) on the end of their shaft.

Fig. 4-3 A milling chuck that uses collets designed for cutters with screwed shanks.

Fig. 4-4 Threaded-shank milling cutters.

The appropriate collet is placed in the chuck and the nose piece tightened just enough to keep it assembled securely, but leaving a

turn or so before it locks. The cutter is pushed into the collet and screwed home. The collet nose piece is then fully tightened, securely gripping the shaft of the tool. This is a popular type of cutter holder; easy to use and capable of accurate work.

THE MOST COMMON CUTTER SHANK SIZES

Table 4-2

Shank diameter (metric)	**Shank diameter (imperial)**
6mm	¼in
10mm	⅜in
12mm	½in

Note that all cutter threads are 20tpi Whitworth form, even for metric cutters.

The collet nose piece is then fully tightened, securely gripping the shaft of the tool. This is a popular type of cutter holder; easy to use and capable of accurate work.

The second type of collet system uses the popular ER range of collets (Fig. 4-5). The holder has a shaft integral with the body of the holder. There is a closing nut that fits the head and has a cunningly designed taper that engages with the front of the collet and acts as a closing mechanism.

Fig. 4-5 An ER collet holder and collets.

In use, the appropriate collet is inserted into the nut by pushing it in at an angle with a slight twisting motion until it clicks into the seat. The rim of the taper in the nut is not circular, but once the collet has been inserted, it will not fall out.

Screw the collet-and-nut assembly on to the head, leaving it quite loose. Slide the tool into the collet, then tighten the nut to make the collet grip the shaft securely. This is a two- or three-handed job, using a spanner on the body of the holder and another on the nut, while ensuring the cutter does not fall out of the collet.

Each ER collet can securely grip a small range of diameters, so they are more versatile than the milling collet holders. Milling cutters are made with a small range of standardized shaft sizes, but one of the advantages of an ER system is that it can easily cope with the whole range of twist drills. That is a considerable advantage where a job requires a sequence of milling cutters and drills to be used, making this a popular collet system.

The ER collets shown have twelve slits per collet (six running from the front of the collet almost to the back, and six running from

the rear almost to the front), but higher-quality collets have sixteen slots (eight front to back, and eight back to front). As the collet is squeezed on to the shaft, the slits close to grip the shaft. There are enough slits to provide sufficient flexibility for each collet to close on a range of diameters.

There are several different standards for the collets in the ER system, each of which will hold shafts with a different degree of concentricity and security. Most of the popular and reasonably priced ER collets do not state to which standard they have been manufactured, so it is safest to assume they are to DIN standard 6499 Form B and, for a 6mm (¼in) cutter, concentric to within 0.015mm (0.0006in). Type ER (standard) should be concentric to within 0.010mm (0.0004in) and the ER ultra-precision collets should be concentric to within 0.005mm (0.0002in). This is a popular system because it is so adaptable.

Some tools are designed to hold replaceable cutter inserts, so these tools are usually made with an integral shaft (Fig. 4-6). As with the various cutter holders, they are normally secured with a drawbar. Cutter inserts come in a range of sizes and shapes, normally to one of the international standards, and can be made from a range of materials, including carbides of various grades. These allow high-speed machining of mild steel, alloy steel and stainless steel with relative ease.

ER COLLET RANGES

Table 4-3

ER range	Size range: metric (mm)		Size range: imperial (in)	
	Nominal	Grip range		
ER 8	1–5 × 0.5mm	0.5–5mm	1/16	0.043–0.062
			1/8	0.086–0.125
			3/16	0.148–0.187
ER 11	1–7 × 0.5mm	0.5–7mm	1/16–1/4 × 32nd	1/32–1/4
ER 16	1–10 × 1mm	0.5–10mm	3/32–3/8 × 32nd	1/16–3/8
ER 20	2–13 × 1mm	1–13mm	5/32–1/2 × 32nd	1/8–1/2
ER 25	2–16 × 1mm	1–16mm	5/32–5/8 × 32nd	1/8–5/8
ER 32	3–20 × 1mm	2–20mm	7/32–3/4 × 32nd	3/16–3/4
ER 40	4–26 × 1mm	3–26mm	1/8–1 × 16th	1/8–1
ER50	6–32 × 2mm	4–32mm		

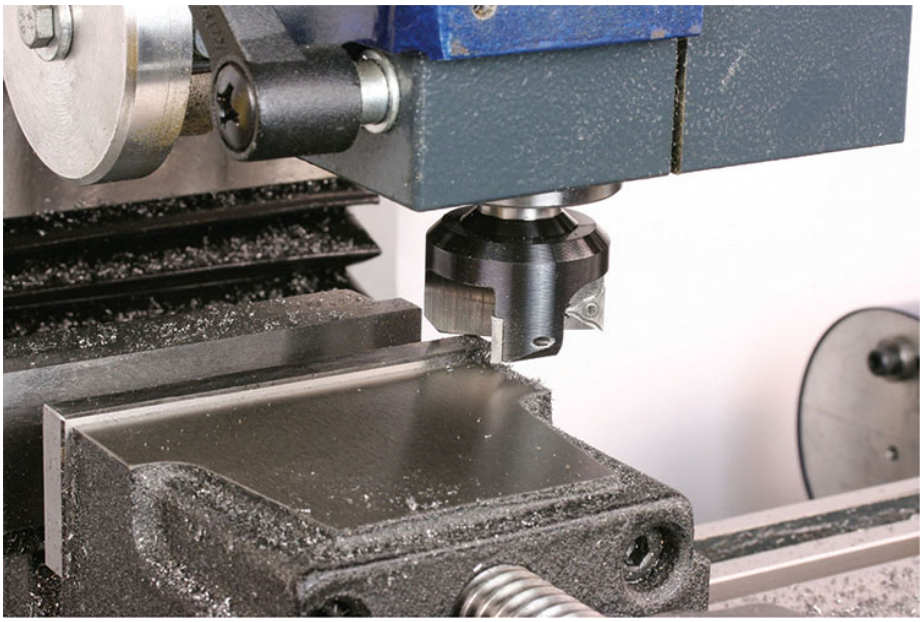


Fig. 4-6 A face mill that uses three inserts as the teeth.

MAKING HOLES

Holes are most commonly made using a drill bit (twist drill), although one of the benefits of CNC machining is that holes of any size can be made using milling cutters.

Twist drills come in a range of diameters and lengths as well as different cutting-edge geometries. The most common twist drills are the ‘jobber’ drills, but ‘extra-long’ drills are sometimes useful. Jobber drills are designed to cut holes of up to three times their diameter, although repeatedly drilling and retracting to clear the drill flutes, and using cutting oil as a lubricant and coolant, allows much deeper holes to be drilled.

Stub drills are shorter than jobber drills, so they are less inclined to flex. These are useful for drilling short holes and for initially drilling a starter hole.

Centre drills (Fig. 4-7), as used in a drilling machine or lathe, are designed to produce an initial shallow hole or mark in material to be drilled. This will give a twist drill a secure start. Otherwise, attempting to drill unmarked material with a twist drill may result in the drill skidding and starting out of alignment with the axis of the hole.

Fig. 4-7 Centre drills.

Chamfering cutters (Fig. 4-8) and countersinks (Fig. 4-9) are designed to chamfer the edge of an existing hole. Countersinks can also be used to create a shaped hole for countersunk-headed bolts. Countersinks come in a range of standard sizes, so they can be used to provide a countersunk hole for standard sizes of bolt and screw heads, including a short section at the top that has parallel sides. Three-flute countersinks work well in most materials.

Fig. 4-8 A 45-degree chamfering cutter.

Fig. 4-9 Countersinks.

Chamfering tools are often used to provide a short chamfer on the edge of a work-piece and can be run along a straight or curved

edge as well as around the top edge of a hole.

Reamers (Fig. 4-10) are designed to finish a drilled hole, producing an accurate size and a fine internal finish suitable for an accurately sized shaft. A reamer has several full-length cutting edges and is run at approximately one third of the speed of a twist drill. For up to 8mm or so, use a drill 0.1mm smaller in diameter than the reamer. For over 8mm, drill the hole 0.2mm undersize.

Fig. 4-10 Hand and machine reamers.

'Hand' reamers have a long taper at the front and can be used in a machine or manually, off the machine, by gripping them in a tap wrench. 'Machine' reamers have a very short taper at the front and are designed to be used in a milling machine, drilling machine or lathe. Machine reamers are more likely to have a tapered shank, while hand reamers have a parallel shank with flats at the end so that they can be gripped by a tap wrench.

To protect their fragile cutting edges, reamers should be turned in a forward direction at all times, even when retracting them from a hole. Like twist drills, reamers should be advanced into a hole and then retracted to clear the swarf, reaming the hole in stages. When using a hand reamer, the tapered portion should pass right through the hole. This means the appropriate tool for reaming a blind hole (closed at the bottom) is a machine reamer, as that is why machine reamers have a very short taper at the start.

Boring tools are specifically designed to create larger holes and to give a fine finish to the bore, which is useful where a large (and expensive) reamer would otherwise be required. Any hole above convenient drill diameter can be brought to size using a boring tool.

For a milling machine, the most useful type of boring tool is the adjustable boring head (Fig. 4-11). The boring bar is held in a slide that can move in or out to set the radius of sweep of the tool tip, so that different sizes of holes can be bored. The minimum diameter of hole that can be bored is limited by the size of the boring bar and tip. There is a trade-off between size and accuracy, because small bars have a much greater tendency to flex than larger-diameter bars, and the thicker and shorter the bar, the more rigid it will be, but that does affect the minimum diameter of the hole and the maximum length of the hole that can be bored.

Fig. 4-11 An adjustable boring head.

In many cases, though, these limitations can be overcome using CNC machining techniques so that, for example, shallow holes with too large a diameter to be bored with a conventional boring head can be created using an end mill. Very small holes can also be created in much the same way, although the rigidity of small milling cutters is affected by their small diameter in just the same way as for small-diameter boring bars.

CREATING FLAT SURFACES

Flat surfaces can be produced using a range of off-the-shelf cutters including end mills, slot drills, dovetail cutters, T-slot cutters, fly cutters, slitting saws and side-and-face cutters.

End mills and slot drills (Fig. 4-12) are useful for machining horizontal or vertical faces as well as grooves or slots.

Fig. 4-12 A slot drill with two cutting edges and an end mill with four cutting edges.

Dovetail cutters (Fig. 4-13) and T-slot cutters (Fig. 4-14) are designed to form particular shapes.

Fig. 4-13 A 60-degree dovetail cutter.

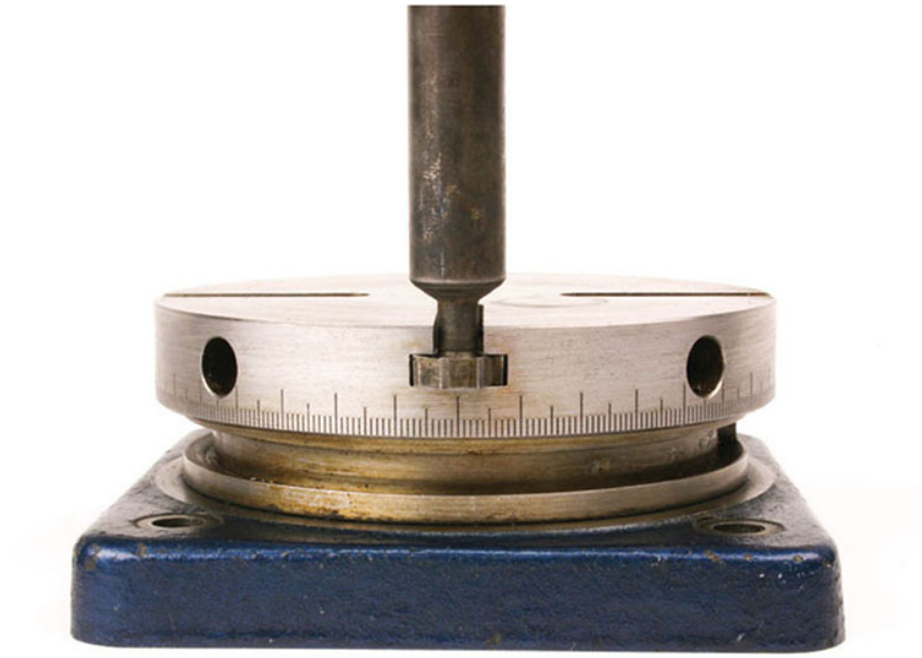


Fig. 4-14 A T-slot cutter in the slot it has cut.

Fly cutters (Fig. 4-15) are single-point tools and are useful for sweeping large surfaces to create flat faces. A typical fly cutter has a generous radius of cut and sweeps a relatively large area compared to a typical end mill or slot drill. Although a CNC mill can easily perform a large number of parallel passes across the face of a workpiece using an end mill, a fly cutter will normally do the same job in fewer passes and do the whole job more quickly.

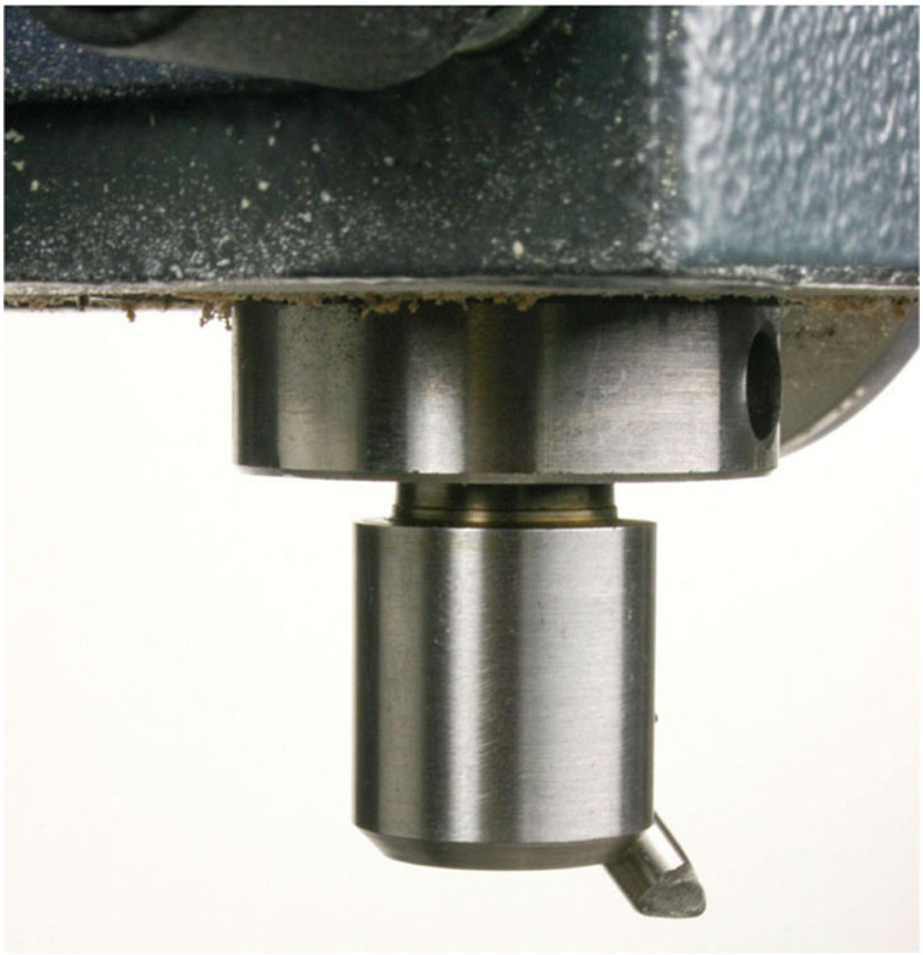


Fig. 4-15 A single-point fly cutter.

A face mill (Fig. 4-6) has some of the characteristics of both an end mill and a fly cutter. The face mill has multiple teeth set around the periphery of a relatively large body (say 50mm (2in) upwards) and acts like a fly cutter, but can cut more material during each pass. This is a useful cutter for larger areas.

Slitting saws (Fig. 4-16) are thin circular saw blades used for cutting slits in collars and slots in screws, and for separating one piece of material from another.

Fig. 4-16 A slitting saw.

FORMING SHAPES

Although the power of CAM and CNC software allows many shapes to be generated using standard cutting tools, there are occasions when it is simpler to create a shape using a form cutter. A shape is generated when it is created by the movement of tool and workpiece, so the resulting shape need not be anything like the shape of the tool. An example of this might be an end mill that is cylindrical in shape yet, with appropriate movements around a workplace, can generate a straight edge on a square workpiece. A shape is formed when the shape of the tool creates the shape of a feature in the

workpiece (or, usually, the inverse of the shape of the tool). An example might be a twist drill that has a cylindrical shape and produces a cylindrical hole. Form cutters behave like woodworking cutters for shapers or routers and reproduce the inverse of the shape of their cutting edge in the workpiece, although there is a more restricted choice of off-the-shelf shaped cutters for metal. Ball-nosed end mills machine a groove with a semicircular bottom (Fig. 4-17); corner-rounding cutters remove a quarter-circle section from the edge of a workpiece to round the edge between two faces (Fig. 4-18); concave cutters leave a protruding ridge on the work (Fig. 4-19); and convex cutters machine a semicircular groove into the work (Fig. 4-20). Although these are useful types of cutter, one of the benefits of CNC milling is that a lot of shapes that would once have been formed using a shaped form cutter can be produced by generating the shape using a conventional end mill or ball-nosed end mill.

Fig. 4-17 A ball-nosed cutter.

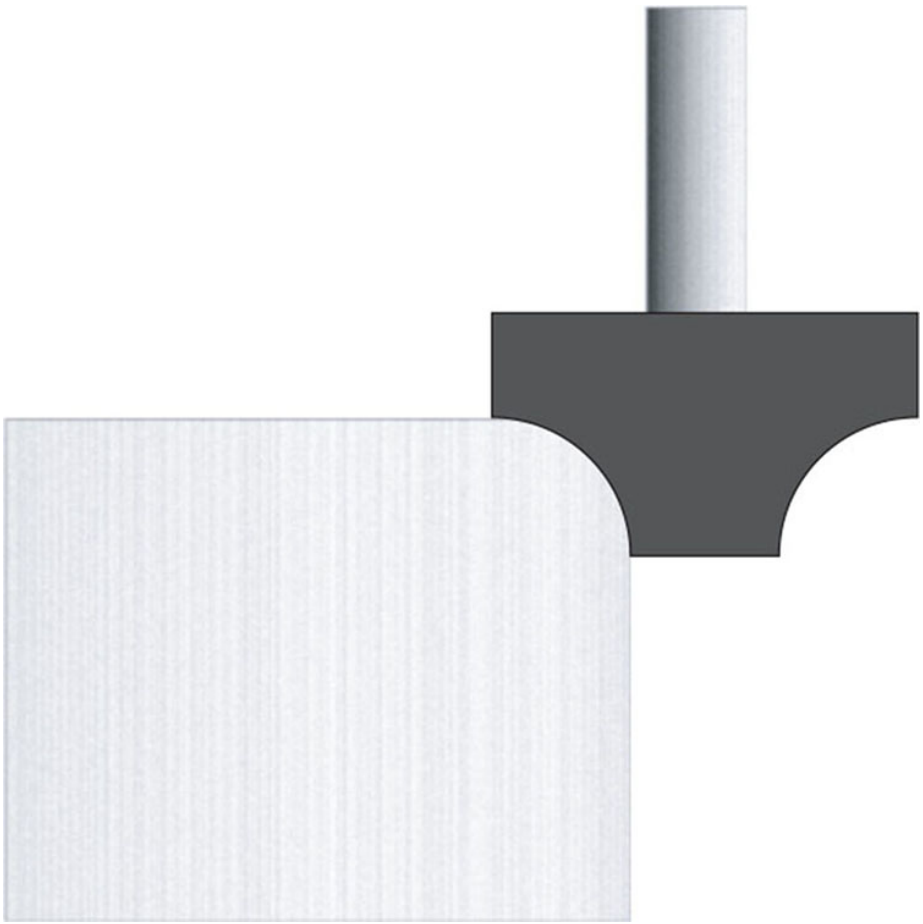


Fig. 4-18 The effect of a corner-rounding cutter.

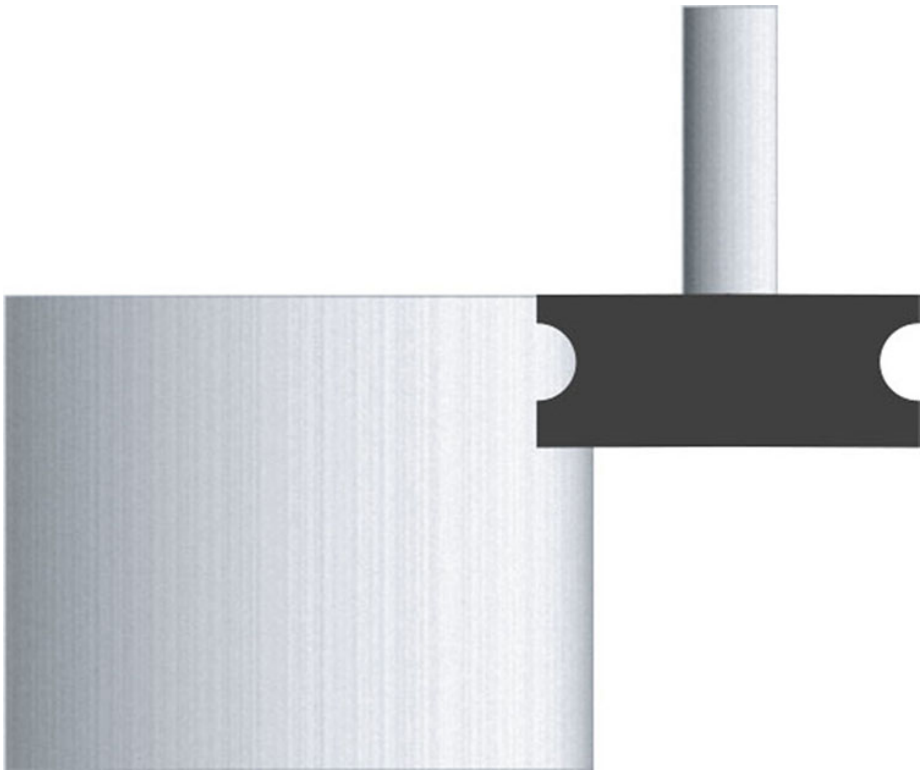


Fig. 4-19 The effect of a concave cutter, leaving a bead on the work.

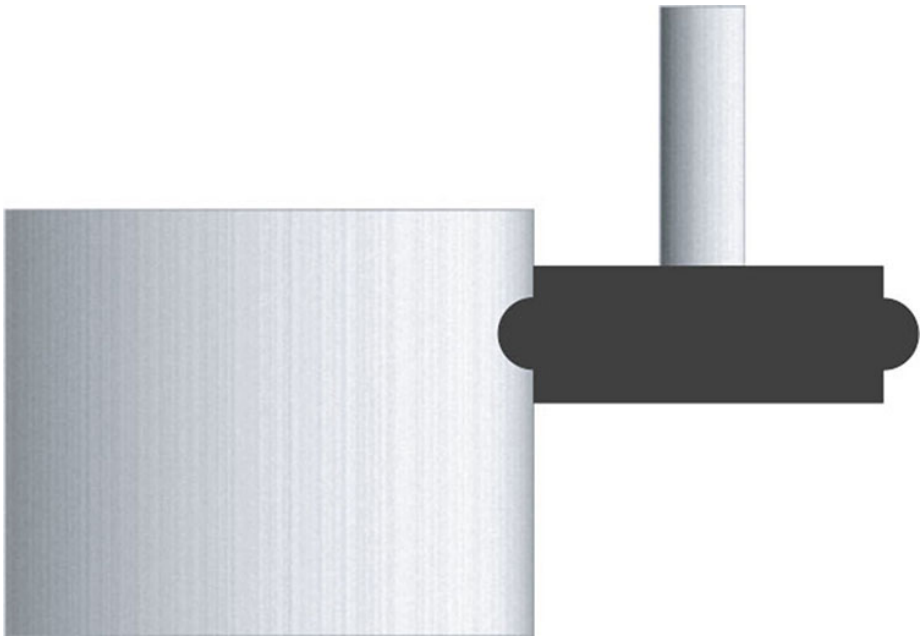


Fig. 4-20 The effect of a convex cutter, creating a groove in the work.

Gear cutters are widely used for forming the shapes of gear teeth, the most common producing involute teeth in a range of standard sizes (Fig. 4-21), while others produce cycloidal teeth as found in clocks and watches (Figs 4-21 and 4-22).

Fig. 4-21 Involute (upper) and cycloidal (lower) gear cutters.

Fig. 4-22 A cycloidal cutter mounted on an arbor.

WORKHOLDING

Some workholding methods will be illustrated throughout the book, but one of the most convenient devices for holding work is a machine vice (Fig. 4-23). Because this is likely to get regular use, it should be an accurate and robust device.

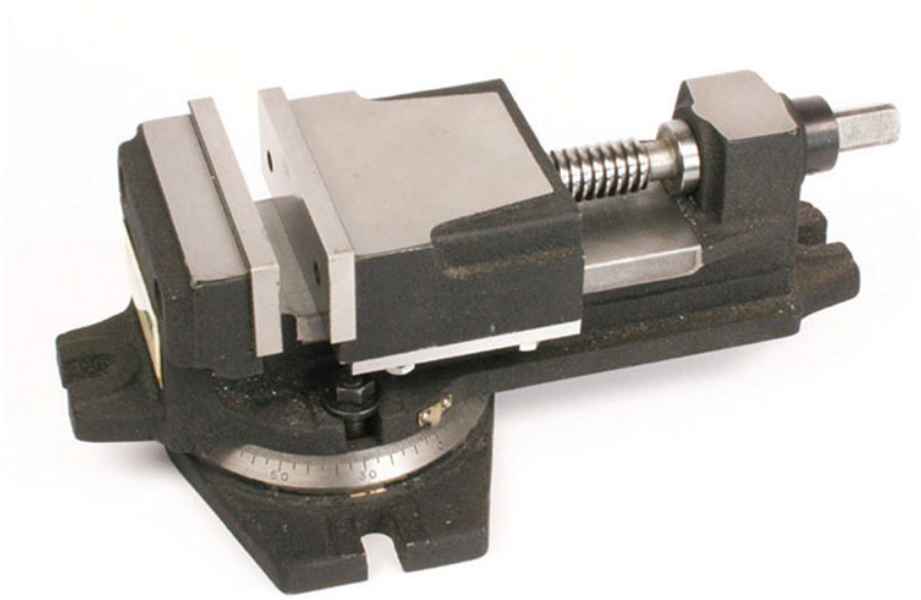


Fig. 4-23 A typical milling vice.

Give some thought to how the vice will be secured to the table of the mill. The vice may come with a rotatable base or a fixed base, but in either case there are likely to be at least two lugs or holes to allow the vice to be bolted to the mill table. Two other useful methods of locating a vice on a mill table are to add a tenon that is permanently attached to the vice and then locates in a T-slot, or to attach a square base plate with two reference edges at right angles and holes for securing bolts (Fig. 4-24). The tenon is a common addition to a machinist's vice, but it limits the mounting orientation of the vice, normally holding the jaws parallel to the X axis. Bolting the vice to a plate allows either reference edge to be located parallel to the Y axis using an engineer's square (Fig. 4-25). The jaws can then be quickly and accurately located parallel to either the X or Y axes to allow the vice to cope with a wider range of jobs.

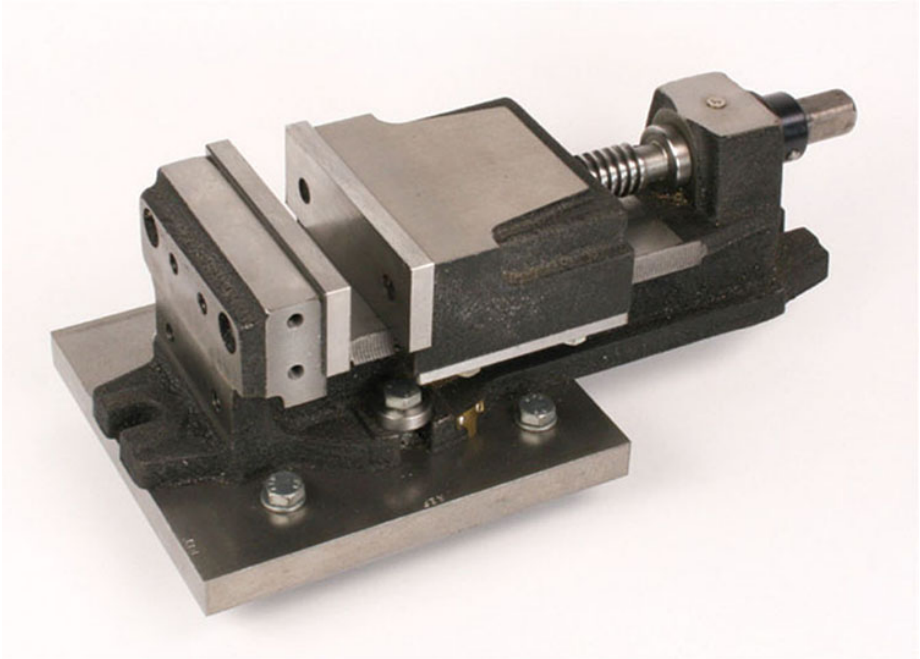


Fig. 4-24 The milling vice aligned on a plate with reference edges.

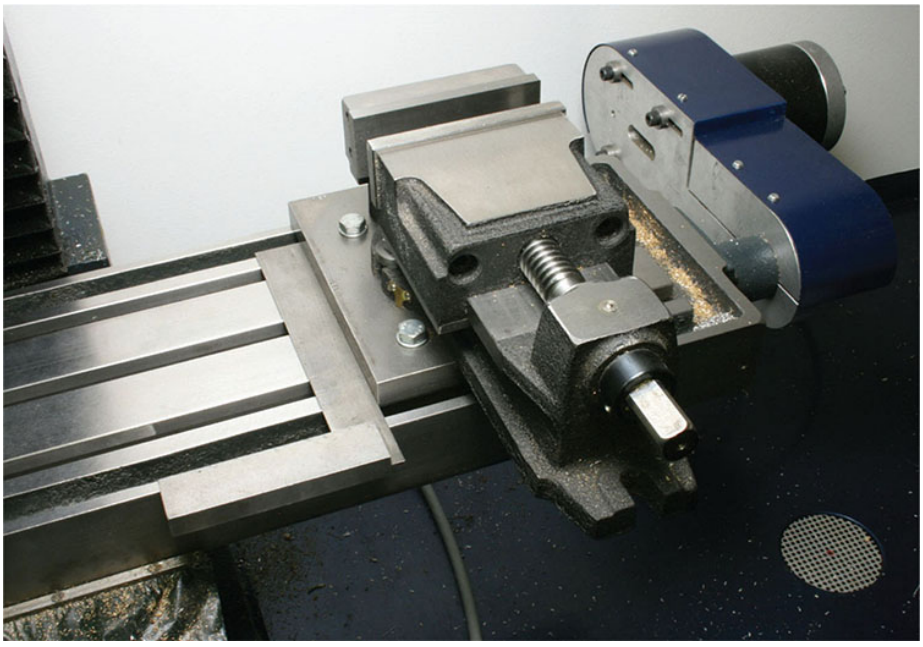


Fig. 4-25 Using an engineer's square to align the reference plate and vice on the mill table.

Most mills have T-slots running along the length of the table and these can be used to secure a vice or a workpiece to the table. Use proper T-nuts if you value your T-slots and your table.

A good investment is a set of T-nuts to fit the T-slots, and some studs, clamps, nuts and washers to go with them (Fig. 4-26). Note that studs must not protrude through the bottom of the T-nuts as there is then a real danger of damage to the T-slots. This is the kind of accessory that is virtually indispensable, but can be made in sizes to suit both the general range of workholding tasks and some specific jobs. For most small mills, the larger ready-made commercial clamping sets are a bit too big.

Fig. 4-26 T-nuts and studs, clamps and spacers.

Work is often held in a purpose-made fixture (Fig. 4-27). The fixture stays on the mill table or in the vice and a series of workpieces is secured in the fixture for machining one after the other. Fixtures are best made for the job in hand, but they are very useful devices for repetition work. There are some examples later on.

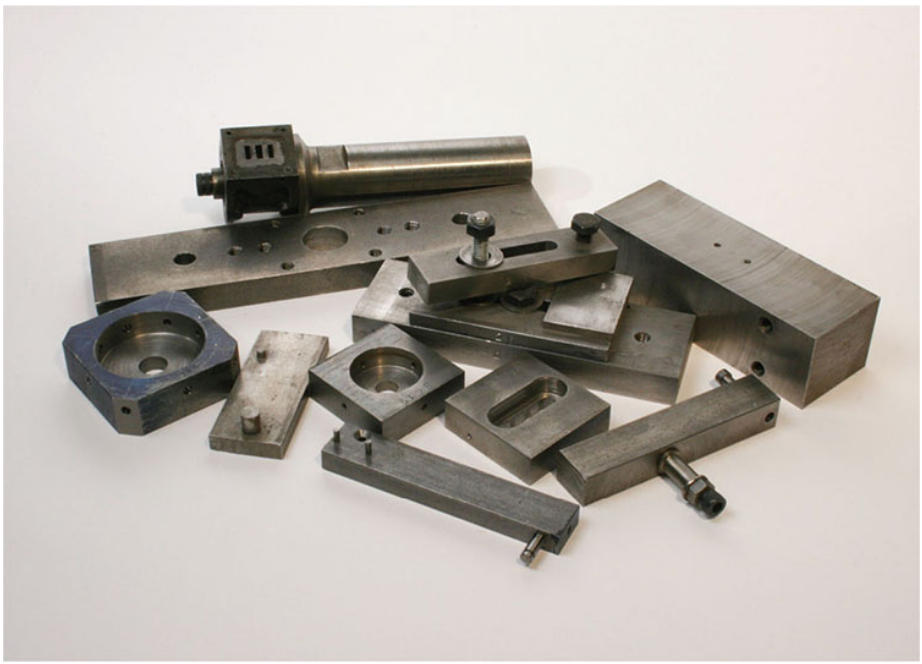


Fig. 4-27 A group of small fixtures.

JIGS AND FIXTURES

A jig is a device for guiding a tool; and a fixture is a device for holding a workpiece securely in a known position. So, for example, you might use a drilling jig that has a bush to guide a drill on to the work (Fig. 4-28). Jigs are not often required on CNC machines, because the machine can position the tool very accurately.

More commonly, you might use a fixture to hold a workpiece securely in a known position on the mill table. Typically, the fixture would be a plate with pins to locate the workpiece accurately and clamps to hold the workpiece in position. Once your CNC program has been homed or touched off so that the position of the fixture is recognized by the program, the work can be machined. Putting another workpiece in the same fixture allows machining to take place in the same relative position on that workpiece. This means fixtures are very useful for repetition machining and are commonly used on a CNC mill.

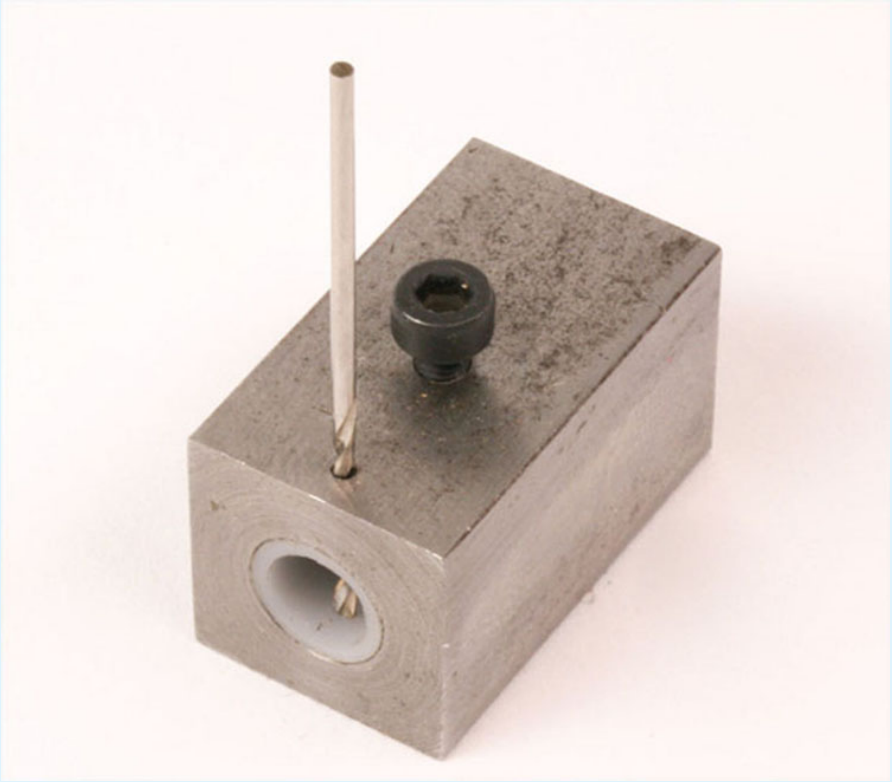


Fig. 4-28 A cross-drilling jig.

SPEEDS AND FEEDS

The spindle speed you should use for a cutter and the feed rate (the speed at which the cutter should move through the work) depends on a number of factors, not least of which is the rigidity of the milling machine. There is a wide variation in rigidity between machines, and the physical size, fit of the slides and condition of the leadscrews will all have an effect. In practice, theoretical speeds that would be ideal for a robust mill in a factory must be reduced as the size and rigidity of the milling machine are reduced.

Ideal spindle speeds, in revs per minute, for small cutters are likely to be unattainable on small machines unless fitted with a high-speed spindle, so there is a compromise here, too. Just get as close as you can to the ideal and accept the compromise in what a particular mill can achieve.

To calculate the spindle speed for a highspeed steel cutter, we need to know the cutting speed appropriate for the material being machined and the diameter of the cutter. Typical cutting speeds are shown in Table 4-4.

Table 4-4

Material	Cutting speed	
	Feet/min	M/min
Hard plastic	500	170
Aluminium	300	100
Brass	200	65
Mild steel	100	30
Stainless steel	50	15

Working in metric units:

- Spindle speed (rpm) = cutting speed (mm/min) / (3 × cutter diameter).

So, for a 10mm diameter cutter machining aluminium:

- Cutting speed is 100m/min or 100,000mm/min.
- Spindle speed = 100,000/(3 ×10), which is 3,000rpm.

Working in imperial units:

- Spindle speed (rpm) = cutting speed (ft/min) × 4 / (cutter diameter).

So, for a ⅜in (0.375in) diameter cutter machining aluminium:

- Cutting speed is 300ft/min.
- Spindle speed = (300 × 4)/0.375, which is 3,200rpm.

Now that the spindle speed has been established, we can calculate the feed rate. This depends on the number of teeth around the periphery of the cutter. Typical end mills or slot drills have four, three or two teeth. The feed rate also depends on the chip load, which is the amount each tooth should cut from the material on each revolution of the cutter. Usable chip loads are in the range 0.125mm (0.005in) for roughing cuts, to 0.02mm (0.001in) for a finishing cut:

- Feed rate = spindle speed(rpm) × no. of teeth on cutter × chip load.

So, for a spindle speed of 3,000rpm, four teeth and a chip load of 0.125mm:

- Feed rate = 3,000 × 4 × 0.125mm/min, which is 1,500mm/min or 1.5m/min.

For a spindle speed of 3,000rpm, four teeth and a chip load of 0.005in:

- Feed rate = 3,000 × 4 × 0.005in/min, which is 60in/min or 5ft/min.

As the spindle speed is reduced, perhaps because the mill spindle cannot reach that speed, so the feed rate must be reduced in proportion, to keep the chip load constant.

For example, a 2mm diameter cutter with four teeth, machining brass, should turn at almost 12,000rpm and have a feed rate of 900mm/min.

Reducing the speed by a factor of 4 to 3,000rpm would mean the feed rate must be reduced to approximately 225mm/min.

In addition, halving the number of teeth to two would mean halving the feed rate, so a 2mm cutter with two teeth, cutting brass at 3,000rpm, would have a feed rate of 112mm/min. Given the fragility of such a small cutter, this feed rate might be reduced a little further in a small machine to somewhere between 50 and 100mm/min (2–4in/min).

Bear in mind that these speeds and feeds are industrial strength and it is likely that they will need to be modified to suit the capabilities of your mill. You may wish to start slowly and work your way up. Experience, and the odd broken cutter, is a useful guide.

A selection of tool-holder blocks with patterns of holes that can easily be created using CNC drilling commands.

5 Linear Programming

In this chapter you will learn how to:

- manually create, edit, save and run a CNC program;
- create a CNC program by using CAM software;
- manually edit a CNC program initially created by CAM software.

Chapter 3 explained how to use commands typed in MDI mode to drill holes at specific locations by following a simple, logical process:

- type modal commands to set the units (millimetres or inches) and various other settings that will apply to all subsequent commands;
- type commands to move to specific positions;
- at each position, drill a hole.

That all works well, but it is rather tedious to have to do that each time we want a particular set of moves. It is also prone to error as we follow a written list of coordinates and type the relevant commands.

Instead of using MDI mode, we can turn our instructions into a simple program that we can create entirely, and preview the results on screen, before running the program with a real workpiece. Saving the program will ensure it can be recalled and run again at any time.

In the CAD/CAM/CNC cycle, simple programming omits the CAD and CAM parts and deals directly with creating a set of G code commands. This is ideal for simple tasks (although simple programs can sometimes do complex tasks) and can be a quick way of producing a part. When we get to the full CAD/CAM/CNC cycle, it is essential that you understand what is happening at the CNC stage, and doing a bit of simple programming is the best way to learn.

There are at least two main ways of creating a program:

- Type commands into a text file then take those into a CNC program. Some CNC programs may allow program commands to be entered directly into the program in ‘program’ mode, then saved as a text file. It amounts to the same thing.
- Use CAM software to create a program. In this case, a drawing is created in the CAM program (or a ready-made drawing, prepared using other software, is brought into the CAM program) and the CAM program creates the program instructions for the CNC program. These instructions are saved from within the CAM program, then imported into the CNC program.

One method is not better than the other; they both have their strengths and weaknesses, but an understanding of both methods is required to be able to make a good choice of method for any job. Both methods are shown below.

CREATING A PROGRAM FROM TYPED COMMANDS

We will create a program to do the drilling for a tool holder like the one you made in Chapter 3, but with different hole sizes and spacings, using a different drilling technique in which the machine does the drilling.

The first stage in creating a program by typing commands is to create a file containing the instructions that make up the program. The best way to do this is as follows.

Mach3: Start Mach3, then choose Edit G code (from the buttons below the program window).

LinuxCNC: It is likely that the LinuxCNC splash screen program will load, so choose File > Edit to start the default

editor. Delete all the existing G code before typing the instructions for your own program.

A file can be created from scratch in any word processor capable of working with plain text (which should be most word processors). On a PC running Windows, it could be something as simple as WordPad; in Linux, it might be gedit. There are some additional steps to take as you save the program, if you are using a word processor, so that the CNC program will be able to read the resulting file.

Begin by typing a few comments that will remind you, later, of what the program does. Comments are words that are ignored by the CNC program but remain visible for you to read. In this book, comments are always enclosed in round brackets.

Mach3: In Mach3, anything inside parentheses (round brackets) is treated as a comment.

Any line beginning with the % character is treated as a comment line.

Anything after two forward slashes // is also treated as a comment, but must finish at the end of that line.

LinuxCNC: In LinuxCNC, anything inside parentheses (round brackets) is treated as a comment.

Any line beginning with a semicolon ; is treated as a comment line.

Type the following two lines (but modify to suit yourself):

(Tool Holder Holes)
(Created 4 July 2012)

In Chapter 3, using MDI mode, we made some assumptions about how the software would behave. We assumed, for example, that it would be working in metric or imperial units, that distances were all measured from a common start point; that the size of movements would not be scaled up or down and so on.

We cannot make those assumptions all the time because the behaviour of the software may be affected by something we did previously that has been retained as a permanent setting and affects the way the software controls the mill.

Instead, we should make sure the software uses a specific collection of settings by typing the commands needed to put the machine into a known state at the start of every program. To do that, type the commands shown below. There are alternative commands if you are using inches that can be used *instead of* the preceding command. Use *either* the command *or* its alternative, not both. Some of these commands may seem puzzling at this stage, but they will be explained shortly.

Table 5-1

Instructions for millimeters	**Alternative instructions for inches**
(Start of initialization block)	
G21 (set units to millimetres)	**G20** (set units to inches)
G90 (use absolute distances)	
G94 (feed per minute mode)	
G92.1 (cancel offsets)	
G91.1 (incremental arc mode)	
G54 (use coordinate system 1)	
G98 (set retract behaviour for canned cycles)	
G49 (cancel tool length offset)	
G40 (cancel cutter compensation)	
G17 (select XY plane)	
G80 (cancel canned cycle motion	

mode)
(End of initialization block)

This short block of commands will be termed an ‘initialization block’ throughout the rest of this book and, while the commands might vary to suit your own needs, the principle is that every program should start with an initialization block so that the CNC software is placed into a predictable state at the start.

This initialization block is repeated in Appendix III towards the end of this book, for handy reference.

A RESCUE FROM STRANGE BEHAVIOUR

The commands in the initialization block provide the basis for a rescue operation should the machine exhibit unexpected behaviour as a result of commands it has been given. Go into MDI mode and type the commands, one after the other. This will set the software and the behaviour of the machine into a known state, and very often remedies the situation.

Now set the feed rate:
F200 (sets the feed rate to 200 units per minute).
Set the spindle speed:
S1000 (sets the spindle speed to 1,000 rpm).
Start the spindle turning:
M3 (starts the spindle turning clockwise, but only if the spindle is under computer control).

Instead of simply moving to a position then moving the Controlled Point downwards to drill each hole, we can use a ‘canned cycle’. This is a single command that carries out several actions. Canned cycles are available for several common operations, and save time and effort at the programming stage.

G83 X~ Y~ Z~ R~ Q~ is a **peck** drilling cycle that moves the Controlled Point to X~ Y~ then moves the Z axis to a maximum

depth of Z~, but it does this in stages of Q~, retracting to a height of R~ to clear the chips after each Q~ move until it reaches the final depth.

So **G83 X27 Y32 Z-20 R1 Q5** will move to (27, 32) then drill a hole to a Z position (depth) −20, in 5mm stages, clearing the chips at the end of each peck by retracting to height R(1). Once it has reached full depth, it will retract the drill and end the cycle.

~TILDE

Throughout this book, the 'tilde' character ~ is used to indicate where a value must to be entered (usually after a letter).

G0 X~ Y~ Z~ means when you enter this command, you must supply numeric values instead of the tilde characters following X, Y and Z (for example, **G0 X20 Y-10 Z-5**). There is no ~ after the G because this command is to be G0. If another number is used, this will make it a different command that will do something quite different.

In manuals for CNC software packages, missing values are traditionally indicated using either the ~ character or the - (hyphen) character. In this book, ~ is used to avoid confusion with the hyphen or the minus sign.

An initial **G98** or **G99** command, used before the cycle begins, determines whether the spindle will retract to the R value (**G99**) or to the original Z value before the cycle began (**G98**). In this case, because **G98** is included in the initial block of commands, the Controlled Point will finally retract to the previous Z height of 25 (see below).

Now enter the commands required to move the Controlled Point and to carry out the drilling operations (*see* Table 5.2).

Prepare to save the file by adding a single line at the top and a single line at the bottom, each containing only a percentage sign %.

Mach3: Save the file, perhaps calling it Tool Holder Holes. When you save the file, make sure you save it as a text file, sometimes called plain text (TXT)

Table 5-2

Instructions for millimeters	**Alternative instructions for inches**
G0 Z25 (to clear all clamps and bolts)	**G0 Z1**
G0 X0 Y0	**GO X0 Y0**
G83 X25 Y10 Z-10 R2 Q3	**G83 X1 Y0.375 Z-0.375 R0.1 Q3**
G83 X50 Y10 Z-10 R2 Q3	**G83 X2 Y0.375 Z-0.375 R0.1 Q3**
G83 X75 Y10 Z-10 R2 Q3	**G83 X3 Y0.375 Z-0.375 R0.1 Q3**
G83 X37.5 Y25 Z-10 R2 Q3	**G83 X1.5 Y0.375 Z-0.375 R0.1 Q3**
G83 X62.5 Y25 Z-10 R2 Q3	**G83 X2.5 Y0.375 Z-0.375 R0.1 Q3**
G83 X25 Y40 Z-10 R2 Q3	**G83 X1 Y0.375 Z-0.375 R0.1 Q3**
G83 X50 Y40 Z-10 R2 Q3	**G83 X2 Y0.375 Z-0.375 R0.1 Q3**
G83 X75 Y40 Z-10 R2 Q3	**G83 X3 Y0.375 Z-0.375 R0.1 Q3**
G80 (cancel canned cycle motion)	**G80**
G0 X0 Y0 Z25	**G0 X0 Y0 Z1**
Stop the spindle:	
M5	**M5**
Then end the program by typing:	
M30	**M30**

LinuxCNC: Save the file, giving it the file extension .ngc

The structure of the program you have created is shown in Fig. 5-1; the main blocks are typical of a straightforward program. It is useful when creating a program to think of this structure and what is required for each of the main sections.

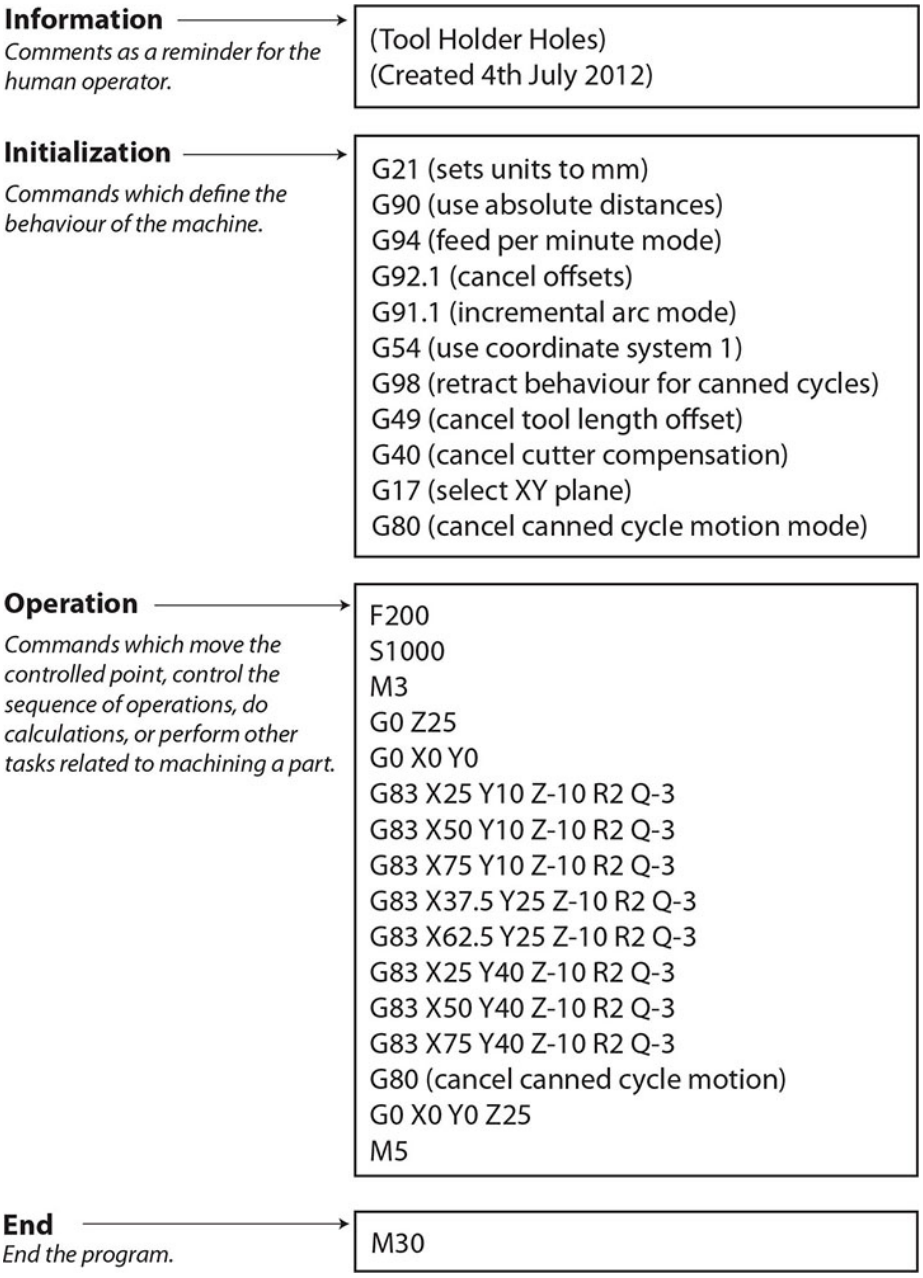


Fig. 5-1 The structure of a simple CNC program.

Start your CNC program, then load the file into the program by using the **File>Load** command.

A preview window should show the path of the Controlled Point, and should display a series of straight line moves and a pattern of hole centrelines where the **G83** commands will cause the program to drill a hole. This is the toolpath (that is, the movement of the Controlled Point required for a specific tool to machine the required shape). Following the lines carefully should show the various moves from the start to each hole in turn, then back to the origin.

Both Mach3 and LinuxCNC allow you to tilt the view of the toolpath either by using the mouse or by clicking on an icon to change the viewpoint. In some views, you will only be able to see the path as moves in the XY plane (as if you were the Controlled Point, looking down on the table). Viewing from another angle should allow you to see the Z movements as well. These Z movements are produced by the G83 drilling commands and they consist of a mixture of cutting feed-rate moves and rapid feed-rate moves in and out of each hole, as the program peck drills each hole.

In Mach3 you will be able to select Offline Mode and run the program without actually moving the real machine.

Check that the predicted path of the Controlled Point is as you expected by examining the toolpath preview window. For this program, the path should consist of horizontal movements from one position to the next and vertical movements representing the drilling operations. Remember that the software is only concerned with the movement of the Controlled Point and has no concept of the shape actually produced by the tool, so it cannot show you the shape of the finished workpiece. This is where you need to use your own imagination.

Put a drill bit into the chuck – 6.2mm or 17/64in would do (that is, a little larger than 6mm or a little larger than ¼in, to provide clearance on shafts of 6mm or ¼in). Put a block of soft material (like wood, plastic or aluminium) at least 100mm (4in) left to right, 50mm (2in)

back to front and at least 18mm deep (¾in) on to the bed and clamp it using two clamps as shown in Fig. 5-2. Take a moment to set the material square on the table, with the sides parallel to the edges of the table, before tightening the clamps. For this job, if the material has been cut reasonably accurately, it can be set with the long edge parallel to the X axis by sighting over the back edge at an angle to allow you to compare the back edge of the work with one of the T-slot edges, just as you did for the project in Chapter 3. It would be a good idea to run this program 'in the air' above the work, without a tool in the chuck, as a visual indication of the way the Controlled Point will actually move.

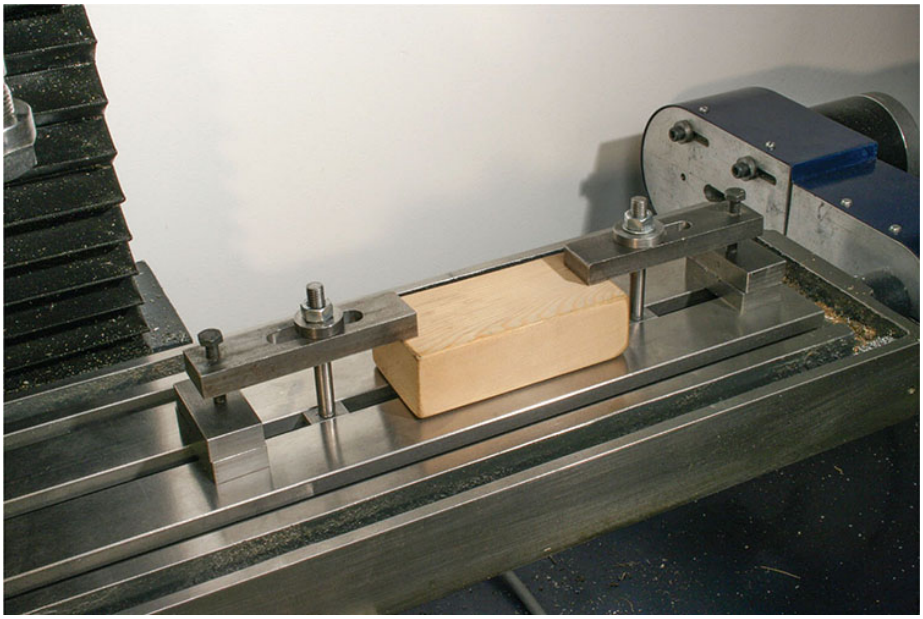


Fig. 5-2 Positions of clamps that secure the workpiece.

Set the origin to X0 Y0 at the front left corner of the workpiece (Fig. 5-3) by touching off the X axis then the Y axis.

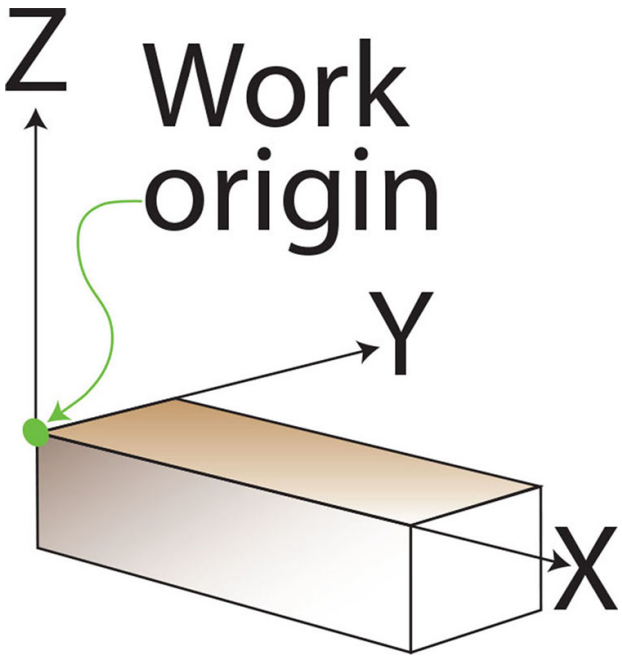


Fig. 5-3 The position of the origin on the workpiece.

Put a drill in the chuck and set the Z origin to Z0 at the top surface of the workpiece by touching off.

Now raise the Z height well above the workpiece and any clamps or other potential obstructions to a height that is easy to remember, bearing in mind that the drill will descend 10mm to the bottom of each hole as the program runs. Set Z to 0 there, by touching off. That way, the machine will perform everything well above the work without actually cutting the workpiece. Now run the program.

To run this program at the proper height to actually machine the workpiece, you will need to touch off Z once again, by going back to that easy-to-remember height you chose, then entering that as the value when you touch off. That way, Z0 should be at the top surface of the work once again.

Check there is nothing that protrudes more than 25mm (1in) (or safe Z) above the top surface of the workpiece (such as clamps or the hold-down bolts for the clamps). If there are obstructions, set the safe Z height to a suitable value within the program so that the Controlled Point can move safely at that height. To do that, you may

need to alter the line that sets safe Z by using an editor, and then reload the program.

Start the mill spindle. If your spindle speed is not under CNC control, set it manually to 1,000 rpm. Now run the program.

As the program runs, it will highlight the command being carried out, so there should be a match between the commands and the movements. The highlight may run one line ahead of the command being carried out at any moment, but the sequence of movements of the Controlled Point should be the same as in the program instructions.

The preview window should also show the path of the Controlled Point as it is being followed, by using a contrasting colour. This is called the Backplot.

Project 5.1

Bed Stop Bar

One of the joys of a CNC machine is that most marking out can be eliminated. A lot of jobs can simply be set up against a fixed stop on the bed of the mill, and for that a simple bar, clamped across the bed of the mill, is a very useful device.

It is most useful if the spacing between the holes is the same as the spacing between the T-slots in the mill table. That way, the bar can be positioned quickly and can also be used as part of a clamping system.

Fig. 5-4 shows a bed stop bar dimensioned in millimetres, but you should amend the dimensions to suit the spacing of the T-slots on your own mill. None of the other dimensions are critical, so they can be varied to suit.

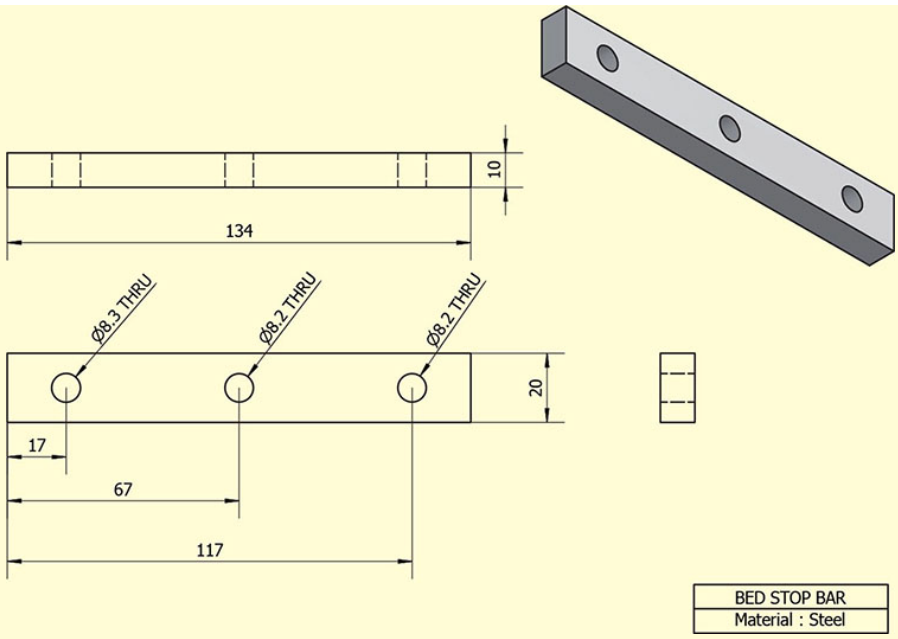


Fig. 5-4 Bed stop bar drawing.

Material

- Mild steel.

Tools

- Twist drill 8.2mm ($^{17}/_{64}$in).
- Centre drill.
- Twist drill to act as a pilot drill, approximately 5mm ($^{3}/_{16}$in) diameter.

Speeds and Feeds

Set the spindle speed appropriate to the drill size. For a 5mm ($^{3}/_{16}$in) drill, use up to 3,000rpm; for an 8.2mm ($^{17}/_{64}$in) drill, use up to 2,000rpm.

Set the feed rate to 50mm/min (2in/min) – but reduce as far as 20mm/min (1in/min) on a very small machine, or increase towards 100mm/min (4in/min) on a larger machine.

Method

- Hold the material in a vice, with the long edge parallel to the T-slots.
- Set the XY origin to the front left-hand corner of the material.
- Set the Z origin at the top surface of the material.
- Set the safe Z height so that the tip of the drill is above the top surface of the material and well clear of any obstructions.

In practice, you may want to use a centre drill, drill through using a pilot drill (perhaps 5mm or $^{3}/_{16}$in in diameter), then drill to final size. On a small mill, you could use an additional pilot drill of 3mm (⅛in). The problem here is that the centre drill should only cut a small depth, but the other two drills need to drill right through the material. In fact, the last two drills need to cut a short distance more than the thickness of the material to make sure the drill tip passes cleanly through the material.

One simple way is to create individual programs, one for the centre drill and one for the other two drills that pass right through the material, then:

- Mount the centre drill, touch off Z, then run the first program.
- Mount the second drill, touch off Z and run the second program.
- Mount the third drill, touch off Z and run that second program again.

The sequence of machining is to move to the first hole position and drill that hole; then repeat those operations for the other holes.

Start your text editor (or go into your CNC program and choose to Edit); then enter your program, save it, load it into your CNC program and run it. A plan for your program might look like:

- Set the cutting speed.
- Rapid move to safe Z.
- Move to the first hole position and drill the first hole.

- Rapid move to safe Z (only required if there are clamps between the hole positions).
- Move to the second hole position and drill the second hole.
- Rapid move to safe Z (only required if there are clamps between the hole positions).
- Move to the third hole position and drill the third hole.
- Rapid move to safe Z.
- Rapid move back to the origin.

Setting the material in the vice

The whole point of this kind of exercise is to eliminate marking out, including not having to mark a centreline along the material.

Chapter 2 explains a range of techniques for finding and setting edges and points, without marking out, and the support website shows how to set a workpiece or a vice accurately, parallel to an axis. Unlike the block for the multitool holder project, sighting against a T-slot is not sufficiently accurate for this job.

If you are using a vice to hold the workpiece, you should set the material level with the tops of the vice jaws to allow clearance below the work for the drill bit to go right through the material without then drilling a hole in the vice.

If you are not using a vice, put a piece of thick material between the mill table and the workpiece, then clamp the workpiece with its long edge parallel to the Y axis. You could do this by choosing an object with a straight edge that is taller than the packing and bar, such as an angle plate or a large block of material, then using an engineer's square with its stock against the front of the mill table to set that object across the table (Fig. 5-5). Clamp the object. Now place the packing and bar against the object, so that the bar will lie parallel to the Y axis, and clamp it. This will allow the drill bit to pass right through the workpiece without damaging your mill table. Position the clamps near the ends of the bar or between hole positions, so that the holes are clear of obstructions. You may wish to remove the reference object

in case the chuck hits it. Check the Y axis travel on the cross slide will allow the Controlled Point to reach the hole positions.

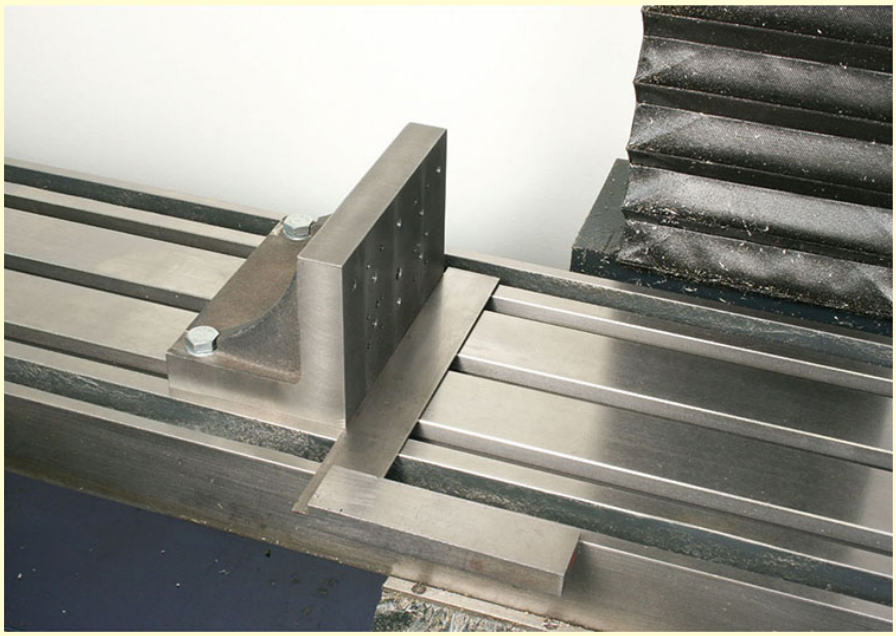


Fig. 5-5 Setting an object square across the table, using an engineer's square.

If you are gripping the material in a vice, you may wish to tidy up the ends of the material by setting an origin somewhere near, but just clear of, one end, then taking some light cuts across that end with an end mill lowered so that it cuts across the full depth of the material. You can set the appropriate Z height by eye. Control the depth of cut by moving the Controlled Point in the X direction. Take enough light cuts until the end is square and clean. If you want to tidy up the other end, use the same technique. This is a purely cosmetic exercise.

Then set the work origin and run the programs. Touching off the Z axis every time you change a drill is a pain, but there are ways around this, as you will learn later.

Once you have made the bar, it can easily be mounted across the bed parallel to the Y axis, using a square as shown in Fig. 5-6,

provided the front of the mill table is accurately parallel to the X axis.

Fig. 5-6 Setting the stop bar across the table, using an engineer's square.

You can then butt work up to the bar, knowing that the edge against the bar is parallel to the Y axis. You can also slide the stock of an engineer's square against the bar, knowing that its blade will be parallel to the X axis. This helps position work accurately on the table.

SAME RESULT;DIFFERENT METHOD

A CAM program can be used to create the instructions for the CNC machine. For a drilling pattern, that means specifying the hole positions for a CAM program.

Using the same size of material and the same size of twist drill should give the same end result, whether programming manually or using CAM software.

Cut2D provides all the facilities we will need for this task. Fig. 5-7 shows the stages in using Cut2D or any similar CAM program.

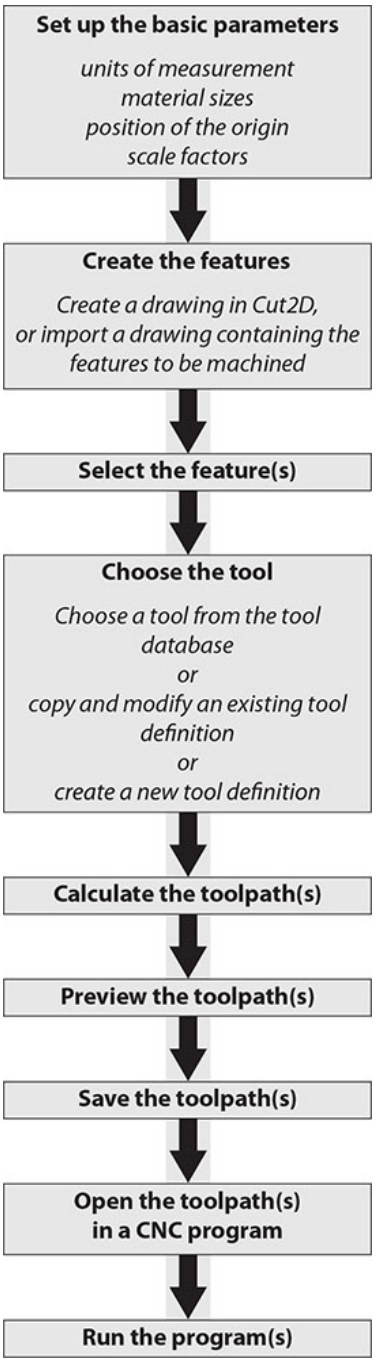


Fig. 5-7 Stages in using Cut2D.

To create a straightforward drilling pattern, run Cut2D and choose to create a new file:

- Towards the bottom left of the Job Setup panel on the screen (Fig. 5-8), choose your preferred units.
- Specify the material size as: length 100, width 50 and thickness 18mm (or 4in × 2in × 0.75in).
- Make the Z Home position at the top surface of the material.
- Select the XY origin position as the front left corner of the material, but do not set any origin offset. Click OK.
- In the Drawing section, use the circle tool to draw circles of diameter 6.2mm (17/64in) at locations shown in Table 5-3.

Table 5-3

Hole	X	Y
A	25	10
B	50	10
C	75	10
D	37.5	25
E	62.5	25
F	25	40
G	50	40
H	75	40

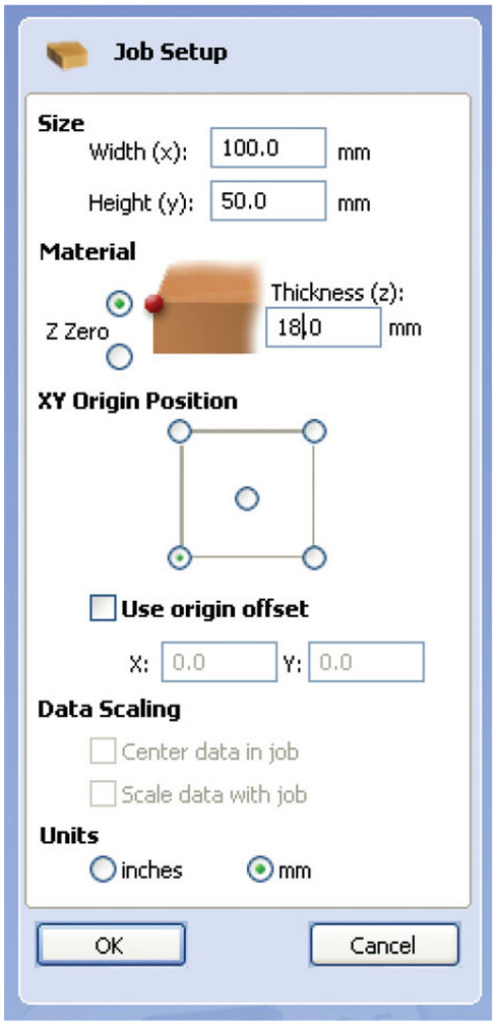


Fig. 5-8 The Job Setup panel.

In fact, the size of the holes is irrelevant for this job. What is important are the coordinates of the centre of each hole, because they locate the position where a hole is to be drilled. If you fit a larger drill, the program will still carry out the drilling operations quite happily. This is important, because it emphasizes that the software has no way of checking what you are actually doing or which tool is in the spindle. Once you have defined the hole positions, shift-click on all the holes in turn to select them all (or drag across all the holes), then go to the Toolpath tab.

Just as when you wrote the G code program by hand you had to specify everything that had to be done, so you must do the same thing when using the CAM software.

Choose Material Setup and set the Rapid Z Gaps Above Material to 25mm (¾in or 0.75in). This is the safe Z height and when the Controlled Point is at this height, it is safe to move anywhere above the surface of the material without hitting clamps or other obstructions. If this is not high enough for the clamps you are using, change the value at this stage. Define the Home position as 0, 0, 25mm (0, 0, 1in). In this context, the Home position is the position that the Controlled Point will move to before beginning the rest of the program; it is also the position the Controlled Point will move to at the end of the program. It performs the same function as the initial movement instruction and the last movement instruction in the manual program above.

Back at the Toolpath menu, indicate that this is a drilling operation by choosing the Create Drilling Toolpath icon:

- Enter a Start Depth of 0. A value of less than 0 would indicate that drilling was to begin at some level below the top surface of the work (like drilling in the bottom of an existing pocket).
- Enter a Cut Depth of 13mm (½in), because the holes should all be that depth.
- Now choose a tool. From the list, choose the nearest drill to the size of the holes (6.2mm or 17/64in). It would be useful to define a new tool of exactly the right diameter, but we do not need to do that now. Let's just get on with the job. Each tool has an associated set of predefined speeds and feeds, so that will take care of both of those settings within the resulting G code.

Peck drilling clears the chips from the drill flutes while drilling. The drill will descend a short distance, retract to clear the chips, go back to where it was, and repeat that cycle as often as is necessary until it reaches the full drilling depth, in a similar way to the G83 canned drilling cycle.

Select Peck Drilling and set the retract gap to 1mm, so that the Controlled Point will sit 1mm above the material when the peck retracts to allow the chips to clear. The peck depth may be set to 6mm (¼in) automatically as a consequence of the choice of tool. The

exact size is not important at the moment. Peck depth is the distance the drill cuts into the work between retracts.

Give the toolpath a name (like Drilling Toolpath) then choose Calculate. The software will calculate the toolpath and list it in a table. The 3D view of the work should show the toolpath (Fig. 5-9) and that view can be rotated to view the path from any angle, in much the same way that a toolpath can be previewed in Mach3 or Linux- CNC.

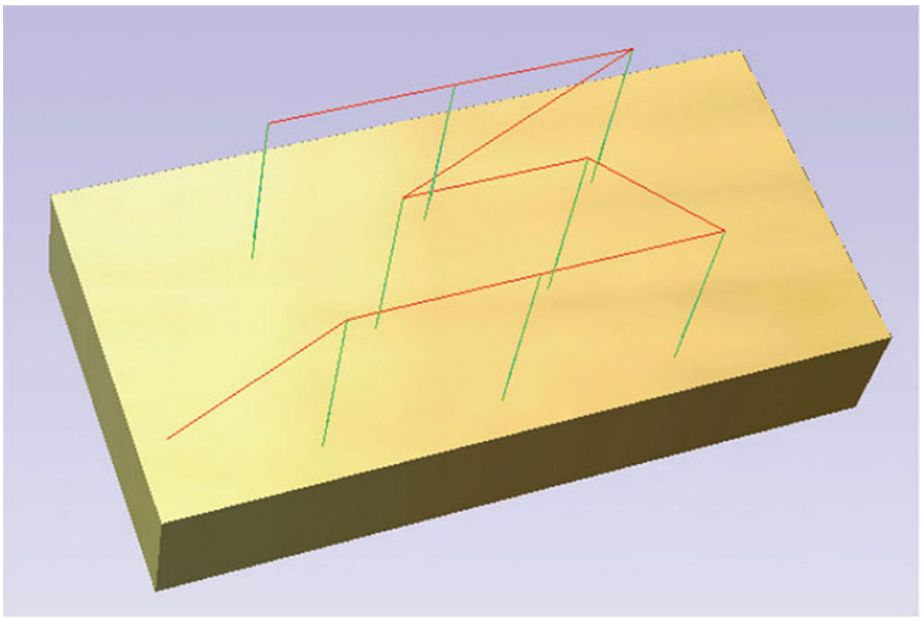


Fig. 5-9 Preview of the toolpath.

Back at the Toolpath tab, Preview Toolpath shows what the workpiece will look like after the program has run.

Then output the G code for that toolpath, in a form suitable for the CNC program:

- Click to highlight toolpath in the list, then go back to the Toolpath Operations menu and choose Save Toolpath.
- Choose the postprocessor that suits your CNC program. For Mach3, choose the Mach3 Arcs postprocessor, in either inches or millimetres. For LinuxCNC, choose LinuxCNC or EMC2. If neither

LinuxCNC nor EMC2 is shown, you should contact Vectric who can supply this postprocessor.
- Enter a name for your file, perhaps choosing the same name as your toolpath, or something similar.
- Save the file.

Now that the G code for the CNC program is saved in a file, you may exit Cut2D.

Start your CNC program, and load the file you just saved. You should see the G code instructions and the toolpath. The toolpath should be the same as in Cut2D, although there is no facility to preview the resulting workpiece. The G code files for Mach3 and LinuxCNC or EMC2 differ slightly, but a careful examination would show that the bulk of the code is the same.

In the Mach3 version, you should be able to identify:

- comments at the start (note that the lines of G code generated by the Linux-CNC or EMC2 postprocessors do not have comments at the start);
- S and F codes for spindle speed and feed rates respectively (if these are the default values for the drill size chosen from the Cut2D tool table, they may be quite high);
- G0 moves at rapid rate;
- G1 moves at cutting feed rate;
- X, Y and Z coordinates;
- the Z moves the peck drilling.

The G code instructions do not use the **G83** canned cycle, but use a sequence of **G1** and **G0** moves instead. In this case, if the peck is 6mm and the hole depth is 13mm altogether, a modified peck distance is used so that the last full peck takes the Controlled Point to the full depth. The sequence is shown as:

G1Z-4.333F1200.0
G0Z1.000
G1Z-8.667F1200.0

G0Z1.000
G1Z13.000F1200.0
G0Z20.000

This is typical of generated code, where there are no extra spaces for ease of readability. The feed rate is specified for every **G1** command and usually for every cutting command. Canned cycles are not used.

The N value at the beginning of each line in the Mach3 file is a line number. The CNC software will ignore this line number, but it is a useful reference for a human being if the CNC software needs to report an error.

LinuxCNC or EMC2 postprocessors do not generate line numbers.

The job can be set up on the mill and the program can be run in the same way as for the manually created G code program. Just take care to follow the same preparatory steps as you did before.

POSTPROCESSORS

A postprocessor is a program that formats the output of CAM software (or any similar program) to ensure that the information is in a format suitable for a specific CNC program.

Project 5.2

Plug Setting Gauge Holding Block

Plug setting gauges are easily made. Each gauge has a head turned to a specific diameter and every gauge has an identical shaft. The gauges are very versatile and can be used, for example, to set a machine spindle a specific distance from an

edge or a step in a workpiece or to locate a hole on a milling machine table or a rotary table.

Gauges can be stored in a block of wood, plastic or metal, rather like a drill holder block, and all the holes in the block will be the same size (Fig. 5-10).

Fig. 5-10 Plug gauges in a holder block.

Design Features

- The hole positions should be chosen to accommodate the sizes of the gauges.
- The holes for the shafts should be a little larger than the diameter of each shaft to allow easy insertion and removal. Too close a fit and there will be an airlock when you insert a gauge. It will also be a little more difficult to remove each gauge.
- The holes should be a little longer than the shafts so that the heads sit down snugly on the block. That is a matter of choice, of course.

Material

- Any close-grained hardwood, a metal such as aluminium or a dense plastic such as Acetal or Delrin can be used to make the block.
- Different designs can be produced, but we will assume this will be a solid block rather than holes in a sheet-metal plate.

Tool

Use a twist drill of diameter slightly larger than the shafts of the gauges. For a 6mm shaft, use a 6.2mm drill. For a ¼in shaft, use a 6.5mm or $^{17}/_{64}$in drill.

Speeds and feeds

Table 5-4

Material	Speed (rpm) [in/min]	Feed rate (mm/min)
Wood	2,000	100 [4]
Aluminium	1,000	50 [2]
Plastic	500	100 [4]

Method

First, choose your material and clamp it to the bed of the mill (Fig. 5-11). That will allow you to measure the highest obstruction and choose a convenient height for safe Z. You can then create a suitable program manually or by using a CAM program.

Fig. 5-11 The workpiece secured to the bed using clamps.

Manual Programming Method

- Set the X and Y origin to the front left-hand corner and the Z origin to the top surface.
- Set the safe Z height above the height of any obstructing clamps.
- For each hole position in turn:
    - Move to the hole position
    - Drill the hole.

Program

<<Initialization block>>

(Remember to include **G98** *to control the retract behaviour at the end of each canned cycle.)*

(Set the feed.)

F100

G0 Z~ *(Retract to safe Z.)*

(Set an appropriate spindle speed – or set it manually.)

(Remember to insert a value instead of ~ – see table above.)
S500
(Start the spindle if the mill has control of the spindle.)
M3
(Starting position.)
G0 X0 Y0 Z~ *(Z height should be Safe Z.)*
(First hole)
G83 X~ Y~ Z~ R~ Q~ *(Remember to insert values instead of ~.)*
(Repeat for the other holes.)
G0 Z20 *(Retract to safe Z.)*
(Stop the spindle.)
M5
(End)
M30

CAM Programming Method

- *Use a CAM program to draw the hole positions.*
- *Set the X and Y origin to the front left-hand corner and the Z origin to the top surface.*
- *Set the safe Z height above the height of any obstructing clamps (in Cut2D this is done in the Toolpath panel by clicking on the Setup Material icon).*
- *Create a toolpath and save it.*
- *Load the toolpath into your CNC program.*

Both Methods

- *Once you can see the toolpath in the Backplot preview window in your CNC software package, touch off X and Y at the left front corner of the workpiece and touch off Z at the top surface.*
- *Check the depth of the drilling toolpath will not exceed the thickness (height) of your material.*
- *Run the program. You might run it in the air above the workpiece first, for safety.*

If you are using a good-looking piece of wood and want to impress visitors to your workshop, give the block two or three thin coats of varnish, rubbing down gently with fine sandpaper between coats. If you do that before you drill the holes, it will prevent the varnish clogging the holes.

When you have finished making the block, give it a quick polish, put some setting gauges in it and place it carefully in a prominent position to invite wonder and admiration. The support website has some examples of how to use these gauges.

BRINGING THE TWO METHODS TOGETHER

Now that you have the code for the drilling pattern and know how to edit a program, you can edit the program created by Cut2D. This is a useful skill as it allows some flexibility in the way we approach creating a G code program for a CNC machine. CAM programs are capable of generating code for complex machining operations, and a knowledge of manual programming and editing would allow some control over the way we use the output of any CAM program.

The easiest way to edit the CNC file from a CAM program is to initiate the editing from within the CNC program. This ensures that the program-type flag (the file-type extension after the end of the file name) remains acceptable to the CNC program. If you choose to edit a program outside a CNC program by using a word processor or a text editor, you should make sure you save the result as plain text, but with a file-name extension to suit the CNC software you will use.

- From within your CNC program, edit the file and add some comments in rounded brackets, then add some blank lines to make the program easier to read. Blank lines can be added simply by pressing the Return key or by inserting a line containing empty comment brackets ().
- Try to separate the code for each hole, beginning with the move to that hole and ending with the retract to **Z20**.

- Then change the feed rate commands, giving them a lower number, for example **F100**. This is easily done using the Search and Replace commands in your editor.
- Then change the spindle speed commands, perhaps to **S1000**. This can be done in the same way as for the F commands.

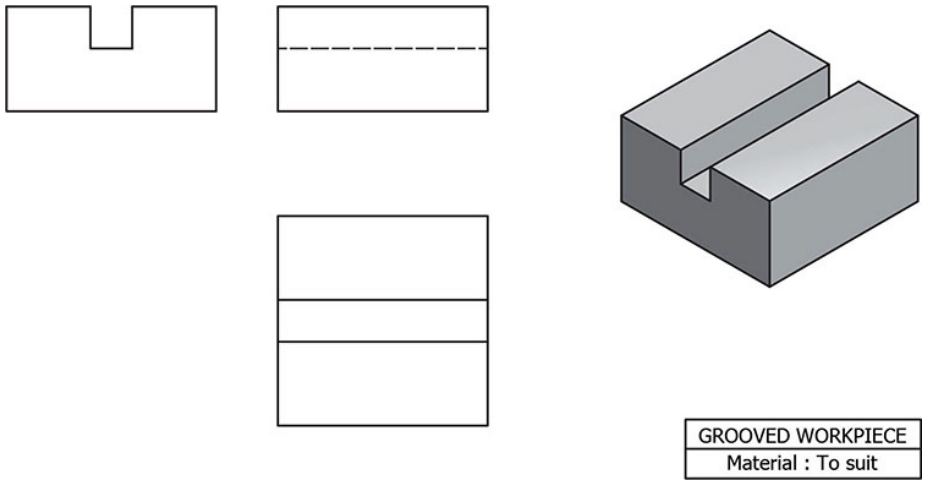


Fig. 5-12 A workpiece with a full-length groove, open at both ends.

This all means that if you have generated a toolpath program using CAM software, it can be quickly tweaked within the CNC program, adjusting it quickly to suit the material being cut (for example, lowering the spindle speed and feed rate for steel as opposed to wood) or the clamps being used (for example, changing the safe Z retract height).

Sometimes, the unforeseen consequences of the position of a clamp or of chuck jaws mean it is useful to be able to edit the motion of the Controlled Point between machining positions. So, for example, it might be necessary to add a move to an intermediate point between two holes in order to move the Controlled Point around a clamp taller than safe Z or a previously machined obstruction that was not taken into account when running the CAM program in the first place. The obvious way to do this is to set safe Z higher, but an alternative would be to edit the toolpath to add G0

commands to move in a different way between holes. Planning a move like this requires considerable care.

TAKING CUTS

Depending on the toolpath and the workpiece, applying a cut is done by lowering the tool then approaching the work, or by lowering the tool while moving across the work. Here are some examples.

To cut a groove right across a workpiece, as shown in Fig. 5-12, start with the tool clear of the work:

- lower the tool (Z axis);
- approach the work using rapid movements;
- cut across the work until the tool is clear of the work;
- move up above the work, to safe Z, at rapid rate;
- move back to the starting position, at rapid rate.

To apply another cut, lower the tool a bit further and repeat the same movements. Repeat that process as often as you need.

That might all appear a bit laborious, but it is easily programmed. Assuming the workpiece is 50 × 50 × 25mm (2 × 2 × 1in) and the work origin XY is set at the front left corner of the work, with Z0 at the top surface of the work; the Controlled Point above (0, 0) and clear of the top of the workpiece; and a 10mm (⅜in) tool in the spindle, the first set of movements might be as shown in Table 5-5.

Apply another cut by lowering the tool further next time. The easiest way to enter this is to copy the commands to cut the groove, paste them after your existing instructions and then change the Z value. It helps, visually, to leave a blank line before pasting the commands into the program.

Table 5-5

Instructions for millimeters	Alternative instructions for inches
<<Type the initialization block	<<Type the initialization block

first>>	first>>
(Move to a safe height.)	(Move to a safe height.)
G0 Z5	**G0 Z0.25**
(Move clear of the work.)	(Move clear of the work.)
G0 X-10 Y25	**G0 X-0.375 Y1**
(Apply a cut.)	(Apply a cut.)
G0 Z-1	**G0 Z-0.040**
(Move up to the work.)	(Move up to the work.)
G0 X-5.1 (just clear of the edge)	**G0 X-0.2** (just clear of the edge)
(Then cut the groove.)	(Then cut the groove.)
G1 X60 (Cut until clear of the end.)	**G1 X2.5** (Cut until clear of the end.)
G0 Z5 (Safe Z)	**G0 Z0.25** (Safe Z)
G0 X-10 Y25	**G0 X-0.375 Y1**

If you need more cuts, repeat the process. So, if you needed ten cuts, you would have ten nearly identical blocks of instructions, differing only by the Z value for the cut.

End the program with **M30**.

Set the work up carefully in the vice. Set the work origin to the bottom left (XY) and the upper surface (Z), and then run the program above the work to check everything (remembering to alter the Z to fool the machine; then reset it before a real run).

Then run the program for real. If you are machining steel or aluminium, use a cutting oil. If you are machining wood, use a dust extractor.

Sometimes, the cut must start within the workpiece. So, for example, a groove that starts inside the boundary of the work (Fig. 5-13) cannot start with the cutter clear of one side of the work. Instead, we must drive the cutter into the work. It is generally a good idea to avoid taking plunge cuts down into the work, if it can be avoided. It is kinder to the cutter and the workpiece to ramp the cutter into the work. End mills that are not centre-cutting cannot be plunged straight down into a workpiece because the centre section of the cutter will

not cut as it descends and it is a lot like trying to push a solid rod through the workpiece. Not a great idea.

To ramp a cut, start above the surface and move along the toolpath as the cut is applied. This is done by starting at one Z height and cutting towards another (lower) Z height. The cutter remains upright throughout the move, but the gradual descent allows the cutter to enter the material, even if it is not designed for centre-cutting.

Assuming the cutter is at X10 Y25 Z0, **G1 X40 Y22 Z-1** will increase the cut by 1mm as the cutter moves from X10 to X40, so the bottom of the groove will be sloped. On the way back, the cutter can remain level (Z not changing) to remove the sloped bottom of the groove.

Further cuts can use the same technique, with blocks of instructions as in the previous example , increasing Z until the last cut is at the final depth.

G0 Z5
G0 X10 Y25
G0 Z0
G1 X40 Y25 Z-1 (ramps into the work)
G1 X10 Y25 (a horizontal cut, back to the start)
G1 X40 Y25 Z-2 (ramps further into the work)
G1 X10 Y25 (and horizontally back to the start)
and so on, to the final depth, then
G0 Z5
G0 X10 Y25

It is not necessary to make the descent last the whole length of the cut, but it saves additional instructions if it does.

USING A CAM PROGRAM TO CUT A PATH

If you intend using a particular CAM program, you should consult the tutorial examples that come with that program so that you have a thorough understanding of the terminology used within that program

and the concepts of paths, profiles and pockets. If you intend using Cut2D, you should visit the support section of the Vectric website and review the video tutorials there. What follows is a very brief review of the basic functions of Cut2D.

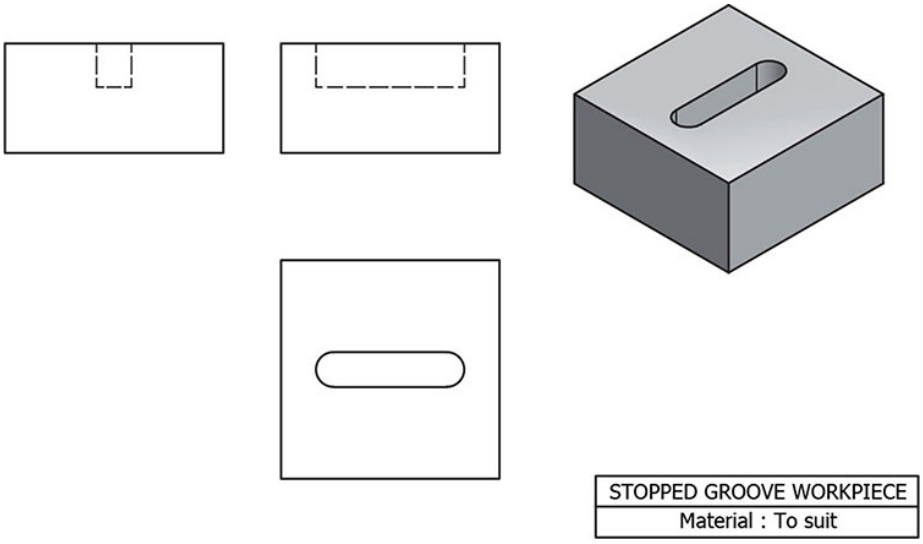


Fig. 5-13 A workpiece with a stopped groove, closed at both ends.

In Cut2D, set up a new job with material 50 × 50mm (2 × 2in) and 25mm (1in) thick. Set the origin to the centre of the material, with Z0 at the top of the material. Then draw the inner 30 × 30mm (1.25 × 1.25in) square shown in Fig. 5-14 by using the Rectangle tool. Make the origin the same as the centre of the first rectangle (25, 25) so that the inner square will be centred within the material. The outline of the inner square represents a cutting path for a square groove.

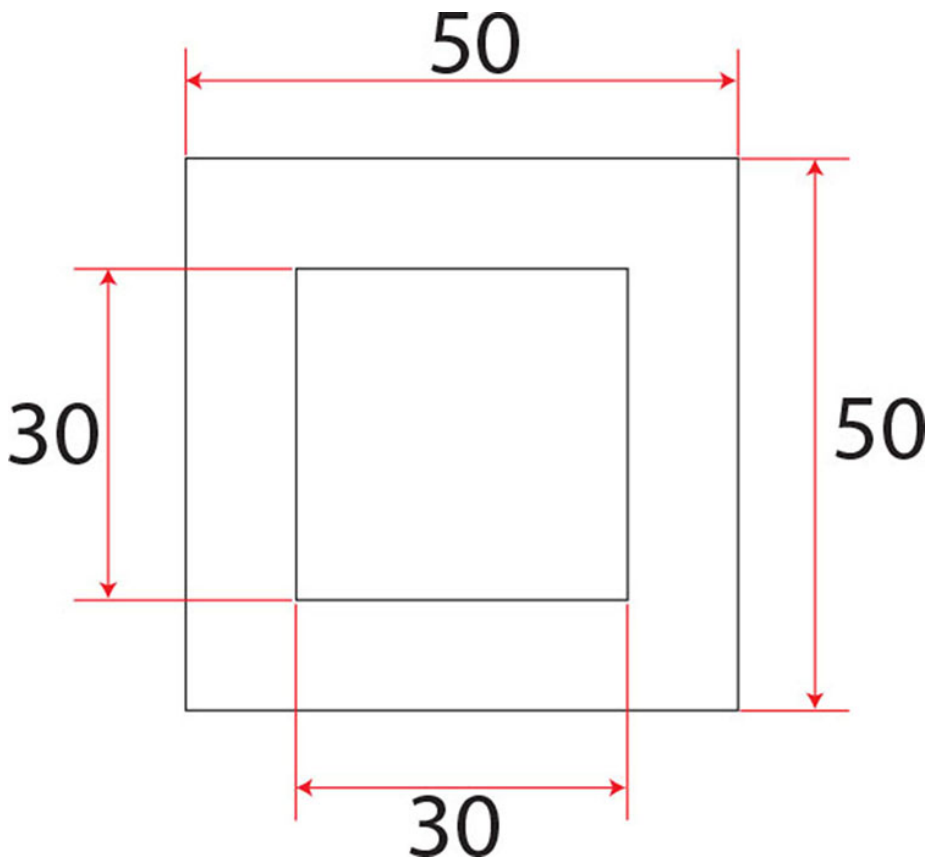


Fig. 5-14 Toolpath for the Controlled Point.

Using the Toolpath menu, choose the Profile Cut tool, then select a cutter the same size as the groove (6mm or ¼in), but make sure you edit it to set appropriate speeds and feeds.

In the Machine Vectors section, select the On Line button. Below that, in the Ramp Plunge Moves section, tick the box to enable this feature and enter a Distance of 20mm (0.75in). Complete the set-up and calculate the toolpath, then preview the toolpath. The square groove will be 30 × 30mm (1.25 × 1.25in) if measured down the centreline of the groove, but will have a larger overall width and breadth across the outside edges because of the width of the cutter. The corners of the groove will be rounded because of the radius of the cutter.

Save the toolpath, transfer the file to your CNC mill software and preview the path there. Remember that, as in your initial drawing, the preview will show the toolpath, not the finished groove. The preview

will omit the outer edges of the square because the software does not know where they are. That is a bonus, because this groove can be cut anywhere on a workpiece if we set the work origin appropriately.

Run it if you like, after first making sure you have set the work origin to the centre of the toolpath (the centre of the 50 × 50 square of material if that is what you are using) and checking that the cutter will not foul the vice or the clamps.

Now try Project 5.3.

Machining a Profile

Fig. 5-15 shows a profile protruding from a workpiece.

- Start by drawing a 50 × 50 × 25mm (2 × 2 × 1in) square in Cut2D.
- Then draw a 45mm (1.6in) square, centred in the material.
- Select the inner square and create a toolpath using the Profile Cut tool.
- Choose a tool wider than the little strip between the inner square and the outer edge of the material; 6mm (¼in) should work well, but anything larger will do the job too.
- Calculate the toolpath, then preview it. The result should be a profile protruding above the remaining material.
- Save the toolpath.

If you want to cut this in the mill, just remember that the cutter will extend beyond the edges of the material, so if the workpiece is held in the vice, the top surface must be clear of the jaws by at least the total depth of cut, plus a safety margin.

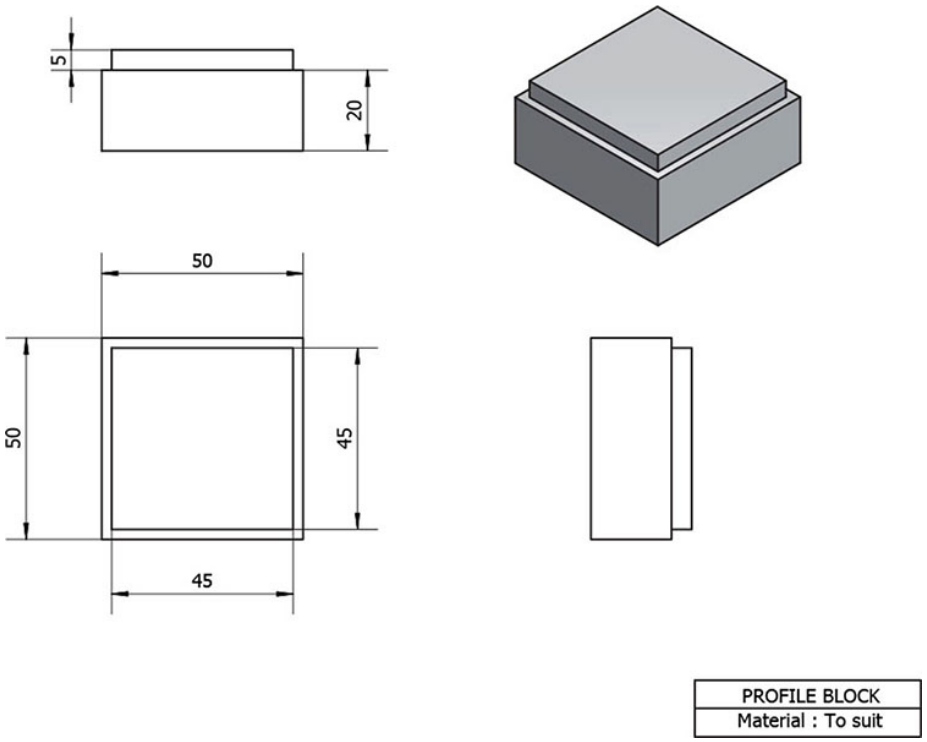


Fig. 5-15 Workpiece with a protruding profile.

Pockets and Profiles

Fig. 5-16 shows a workpiece with a raised profile and a sunken pocket. There is also a lot of material to be removed outside the raised profile. These are common types of feature and each type can be machined using a standard technique. Profiles are raised above a surface and pockets are sunken features.

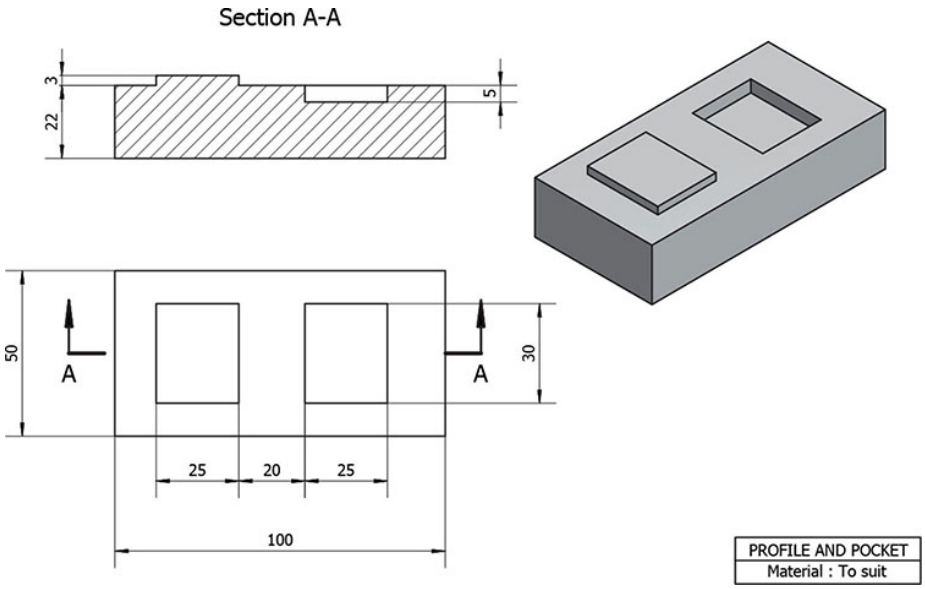


Fig. 5-16 Workpiece with a protruding profile and a recessed pocket.

An area can be cleared using overlapping passes of a cutter, such as an end mill, or a larger cutter such as a fly cutter or a face mill. However, unless you want to do a tool change partway through the job, the ideal cutter for the recessed area will have a small diameter so that the corners have as small an internal radius as possible. There are two approaches.

The first is to do the job in two parts, using a separate program for each part, with a tool change in between. Use a large-diameter cutter to create the profile by clearing the whole area except the profile, then change the tool and run a program that uses a different toolpath to machine the pocket.

The second is to choose a cutter that will be satisfactory for the pocket and use that for the whole job. It will mean more passes of the cutter to remove the excess material when creating the upstanding profile, but that is no great hardship unless you are in a hurry.

For this example, we will use the second method because it highlights an important aspect of the job.

To machine a profile, run the cutter around the outside of the shape, taking account of the diameter of the cutter so that the final

shape is the right size. This takes no account of the material lying outside the profile, though, so it leaves us with a problem on this job. Try selecting the left-hand square, calculating a profile toolpath using a 6mm (¼in) cutter on the outside of the square, then previewing that toolpath. Yes, it will create a profile, but the end result is probably not quite as you expected.

It is all about how you think about the workpiece and the cuts required to produce the end result. On this job, the Profile Cut is not appropriate.

Instead, think of the job like this: the profile is an island within a large area that needs to be removed, so it is an island within a larger pocket. The outer edges of the pocket will be the outer edges of the workpiece (or, better still, a little beyond the outer edges, to make sure the cutter does not leave a thin feather at the edges). So the first stage is to draw a larger rectangle just outside the outer edges of the workpiece, then:

- Select that outer rectangle, shift-select the inner profile and create a pocket toolpath 3mm (⅛in) deep.
- Preview the toolpath to see that the tool cuts away everything inside the pocket until it meets an obstruction, which it works around, creating the profile.
- Now select only the right-hand rectangle, which will be a pocket.
- Create a pocket toolpath for this. For speed, set the Start Depth to 3mm (⅛in) to avoid cutting fresh air and the Cut Depth to 5mm ($^3/_{16}$in).
- Preview all the toolpaths. The result should be as shown in Fig. 5-17.

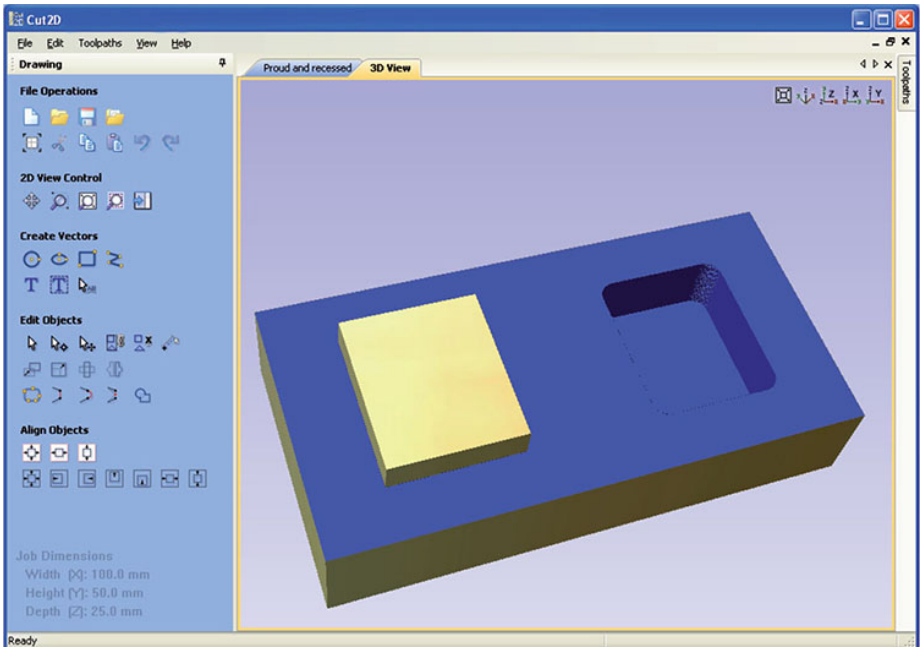


Fig. 5-17 Preview of the toolpath for a workpiece with a protruding profile and a recessed pocket.

The toolpath for this job consists of straight line movements that are not particularly complex. The same principles apply, though, even if the feature has a more complex shape. Fig. 5-18, for example, shows a component that will require several non-linear (that is, curved) paths. The principle, though, is just the same. Working out the paths is a lot more complicated.

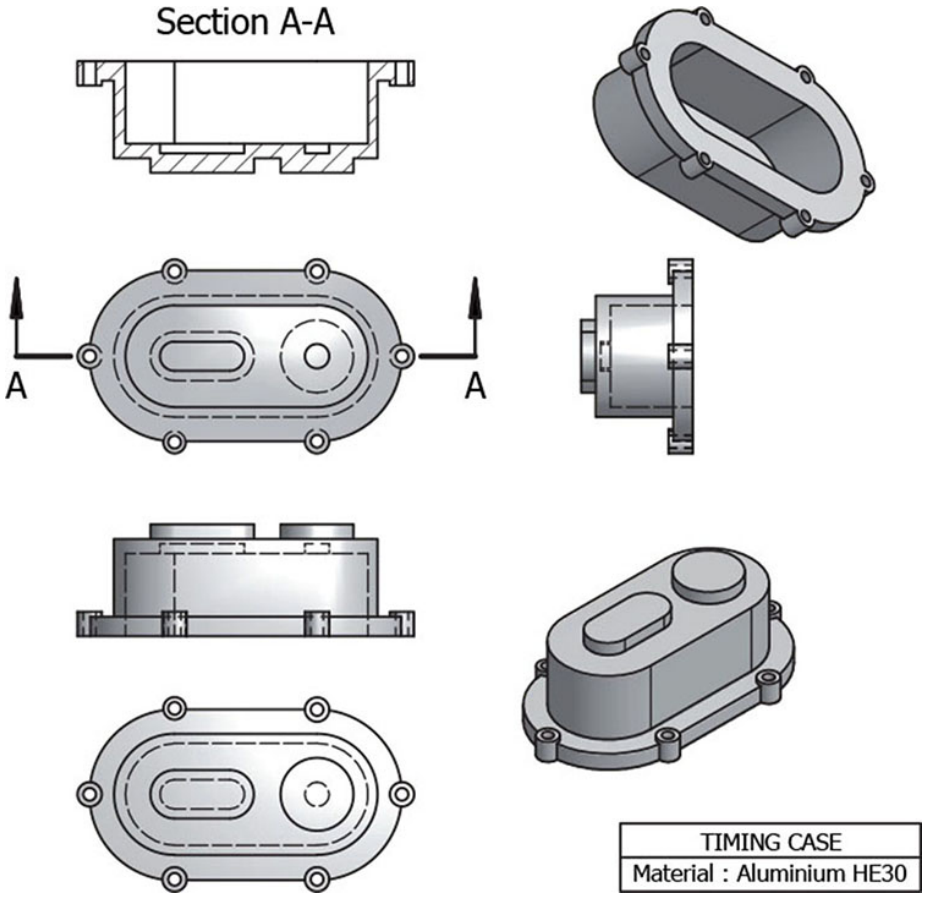


Fig. 5-18 An engine timing case.

For simple areas, profiles and pockets, it is easy enough to create your own program. For more complex shapes, it is quicker and easier to use a CAM program like Cut2D, VCarve Pro or similar. Simply draw the shape in plan view, then create the toolpaths one by one, taking care with the selection of profile or pocket toolpaths.

It all depends on how you look at the problem.

Project 5.3

Ribbed Tool Tray

Keeping the work bench tidy can be a challenge, and the same applies to the space around a machine tool. If there are three or

four tools in use, we need a place to keep them handy but tidy. A simple tray would do, but the tools tend to roll around and get mixed up, and there is a danger the cutting edges may get chipped. A ribbed tool tray is more effective and can be made from wood, plastic or metal. Fig. 5-19 shows a simple tray and Fig. 5-20 shows some suggested dimensions for the toolpath. Note that this is the path of the Controlled Point and the grooves will extend half their width on either side of this path.

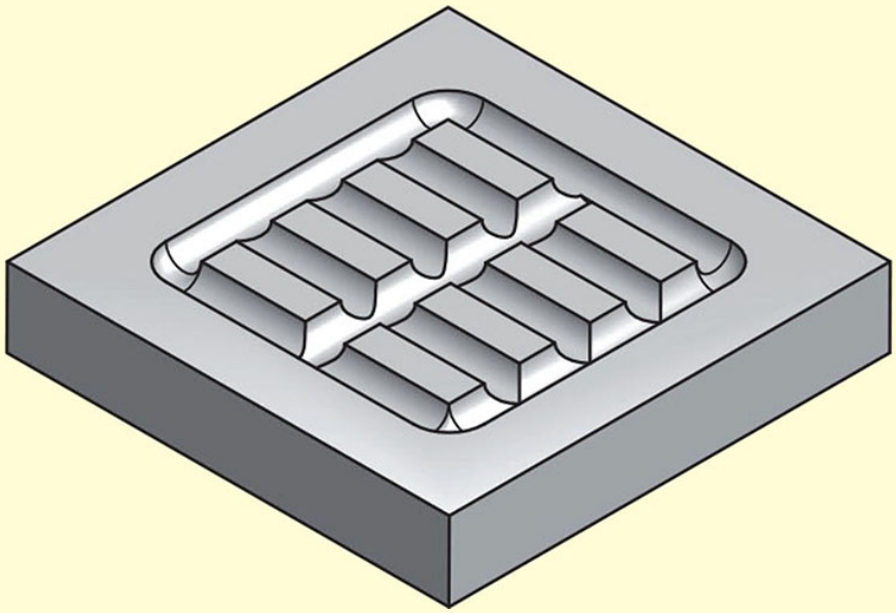


Fig. 5-19 A ribbed tool tray.

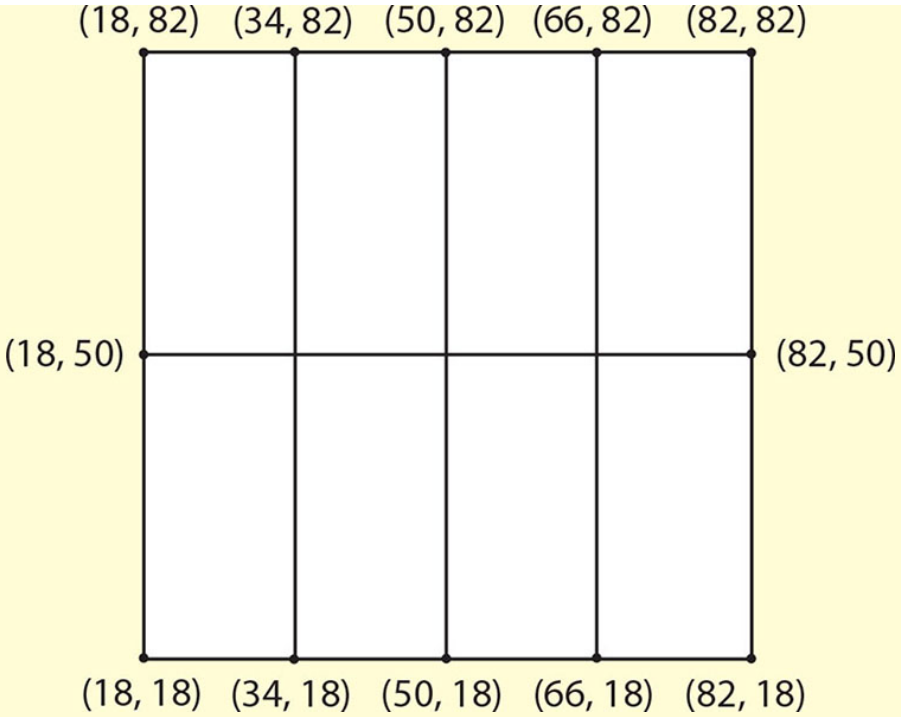


Fig. 5-20 Suggested dimensions for the toolpath.

Design Features

- *The outer groove and the mid-groove running across the tool channels allow the fingers to grip the individual tools.*
- *Widths and depths of grooves and channels are not at all critical.*
- *The shape of the grooves and channels is not critical either, and rounded bottoms or Vee channels work equally well. The outer grooves are best with rounded bottoms or flat bottoms.*

Material

- *Any medium-hard plastic, like Acetal or Nylon, is the best choice, but varnished wood looks nice; or a block of aluminium can be machined quite quickly.*

Tool

- *A round-nosed (ball-nosed) slot drill or end mill, or a Vee tool.*
- *An 'ordinary' square-ended end mill or slot drill would do, although the resulting flat bottoms to the grooves will allow the tools to roll around a little.*

Speeds and Feeds

Using an 8mm ($^{5}/_{16}$in) ball-nosed cutter, Table 5-6 gives the appropriate speed and feed rates.

Table 5-6

Material	Speed (rpm)	Feed rate (mm/min) [in/min]
Wood	5,000 or more	250 [10]
Aluminium	3,000	250 [10]
Plastic	500	100 [4]

Method

- *Set the X and Y origin to the front left-hand corner and the Z origin to the top surface.*
- *Set the safe Z height above the height of any obstructing clamps.*
- *Machine the outer groove.*
- *Machine the mid-groove running across the channels.*
- *Machine the tool channels. When machining the tool channels, move to the front starting position of the channel, lower the cutter and cut to the rear of the channel, then retract.*

Remember that your instructions move the Controlled Point and not the outer edge of the tool, so take care when positioning the Controlled Point at the end of each groove.

A plan for the program might be:

- *<<Initialization block>>*
- *Set the feed.*

- *Set the spindle speed – or set it manually.*
- *Go to safe Z.*
- *Go to the starting position.*
- *Machine the outer groove.*
- *Machine the cross-groove.*
- *Machine the tool channels.*
- *End.*

Whether you do this by machining each groove to full depth then machining the next groove or by machining all the grooves to the same depth then machining all the grooves to the next depth, and so on, is up to you. You could always make two blocks and try both methods. Or make half a dozen and travel round the paths as you please.

Project 5.4

Strap Clamp

A well-equipped workshop can never have enough clamps! Fig. 5-21 shows a simple adjustable clamp, dimensioned in millimetres (and inches), but you should amend the dimensions to suit your own needs.

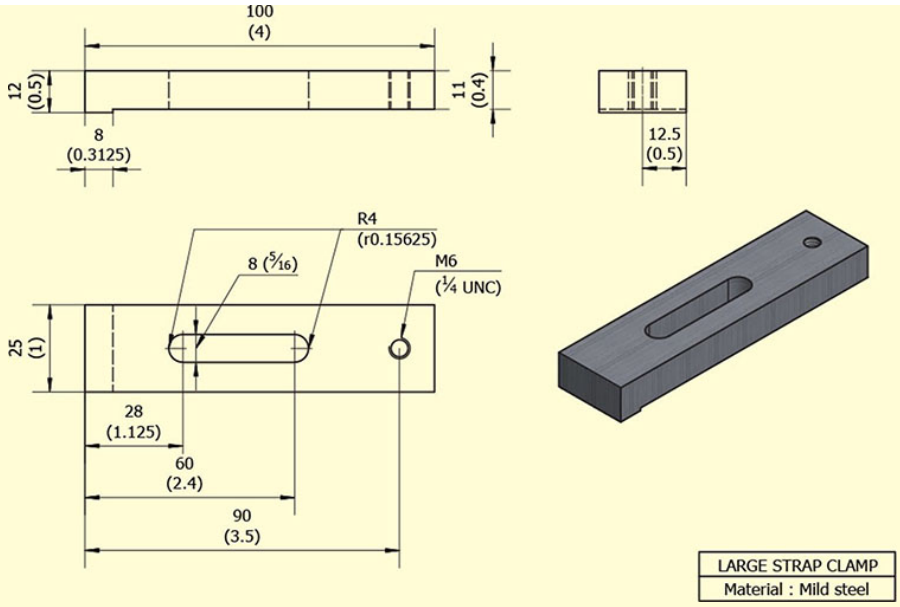


Fig. 5-21 A strap clamp, with suggested dimensions.

The slot should be cut using a centrecutting slot drill or end mill with a diameter just slightly larger than the studs that will fix the clamp. For an 8mm ($^{5}/_{16}$in) threaded rod, an end mill of that same size may provide just enough clearance, or you may wish to use a 9mm ($^{11}/_{32}$in) end mill instead. None of the other dimensions are critical, so they can be varied to suit.

Material

- Mild steel

Tools

- Centre-cutting end mill or slot drill of 8 or 9mm ($^{5}/_{16}$ or $^{11}/_{32}$in) diameter.
- Centre drill.
- Twist drill 5mm (for M6 thread) or a diameter to suit any other suitable thread size.

Speeds and Feeds

- For drilling, set the spindle speed appropriate to the drill size. For a 5mm (¼in) drill use up to 1,500rpm with a feed rate of 90mm/min.
- For cutting the slot and the recess, set the spindle speed to 1,000rpm.
- Set the feed rate to 75mm/min (3in/ min) – but reduce as far as 50mm/min (2in/min) on a very small machine, or increase towards 150mm/min (6in/min) on a larger machine.

Method

- Hold the material in a vice, with the long edge parallel to the T-slots and with the upper surface above the jaws, to allow the recess to be cut without hitting the jaws. In that position, think of the top surface of the material as being the bottom surface of the clamp. All the work can be done from this side.
- Set the XY origin to the front left-hand corner of the material.
- Set the Z origin at the top surface of the material.
- Set the safe Z height so that the tip of the twist drill is above the top surface of the material and well clear of any obstructions. The twist drill is longer than the centre drill or the end mill, so that safe Z will work for all three tools.

Using Your Own Programs

Create four programs:

- The first will centre drill for the threaded hole.
- The second will drill tapping size through the hole.
- The third will face off the recess, using a pattern of passes to clear the whole area and leave a straight edge where the recess meets the front foot.
- The fourth will mill the slot.

Using Cut2D

Follow the stages outlined earlier in Fig. 5-7:

- Draw the job in Cut2D.
- Create the area to be recessed, the slot and the hole for the jacking screw.
- To draw the slot, draw a circle at each end, then draw a rectangle overlapping the two circles (Fig. 5-22).

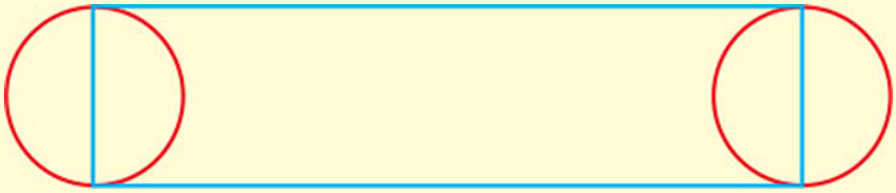


Fig. 5-22 To draw the slot in Cut2D, draw two circles (shown in red) and a rectangle (shown in blue), then join them using the Weld Selected Vectors tool.

- Select the rectangle and both circles, then use the Weld Selected Vectors tool to create the outline of the slot.

Then create four toolpath programs:

- The first will centre drill for the threaded hole.
- The second will drill the tapping size through the hole.
- The third will face off the recess, using a pattern of passes to clear the whole area and leave a straight edge where the recess meets the front foot.
- The fourth will mill the slot by ramping the cutter into the work.

Transfer the files to your CNC program and check the toolpaths are as you expected.

Running the Programs

- Mount the centre drill, touch off Z, then run the first program.

- Mount the tapping-size drill, touch off Z and run the second program.
- Mount the end mill, touch off Z and run the third program to create the recess.
- Run the fourth program to machine the slot.

There are many curved parts in a clock and these are easily produced using CAD/CAM/CNC techniques.

6 Arcs, Circles and Polylines

In this chapter you will learn how to:

- use G code commands to create arcs and circles;
- use CAM software to create a program containing arcs, circles and polylines.

Linear moves are the basis of much machining, but the power of CNC really comes into its own with work that contains curves.

Curves can be circles or part circles, and these are easily dealt with using G code commands or by using a CAM program. But curves may be much more complex, with sweeping arcs and varying radii, and those are best dealt with using a CAM program alone, as the geometry of complex curves means it is both difficult and time-consuming to specify the individual start, end and radius of what may be dozens or hundreds of individual segments. Life is just too short.

SIMPLE CURVES

Simple curves are those that consist of whole or part circles, where the start point, end point, centre and radius of each arc or circle are known.

The **G2** command moves the Controlled Point clockwise around an arc.

The **G3** command moves the Controlled Point anticlockwise around an arc.

The format of both commands is the same: specify the X, Y and Z values at the end of the arc, and the X and Y distances from the

current (start) point to the centre of the arc. Because X, Y and Z values represent the end of the arc, the letters I and J are used for the X and Y distances from the current (start) point to the centre of the arc, and a suitable command for the path shown in Fig. 6-1 would be: **G3 X-5 Y42 Z10 I-10 J-24**, which makes the Controlled Point travel anticlockwise in an arc from the current point (15, 42) to X-5 Y42 Z10.

From the current (start) point to the centre of the arc (5, 18) is:

$X_{centre} - X_{centre}$ (5 − 15 = −10)

so **I−10** specifies the required distance along the X axis.

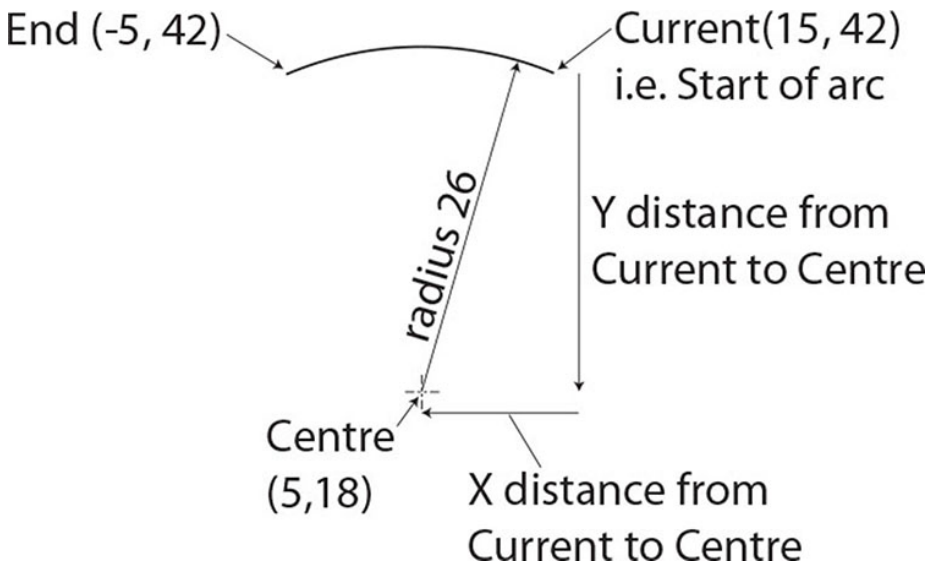


Fig. 6-1 A circular arc.

$Y_{centre} - Y_{centre}$ is 18 − 42 = −24 so **J−24**

specifies the required distance along the Y axis.

The Z height will change from the current Z to the end Z as the Controlled Point moves from the start of the arc to the end.

INCREMENTAL AND ABSOLUTE ARC MODES

There are two different modes that can be used for specifying arcs, **incremental** and **absolute**. The calculations here assume

incremental mode, where I and J values are increments representing the distances from the current point back to the centre of the arc.

In absolute mode, I and J values are the X and Y coordinates of the centre of the arc. Note that absolute mode for arcs does **not** use absolute machine coordinates. The choice of incremental or absolute modes is set by the following.

Mach3: Using the Configure > State menu at set-up.

LinuxCNC: The G90.1 ***command sets absolute arc mode.***
The G91.1 ***command sets incremental arc mode.***
Unless G90.1 ***is used, the system assumes*** G91.1 ***is in operation, but it is wise to include a*** G91.1 ***command at the start of every program.***

In many cases, the Z height remains the same throughout so that the cutter runs in a 'flat' circle, but if the value of Z changes, the movement will be a part-spiral, with the cutter moving down (or up) as it runs around the circle. Interestingly, a thread follows a spiral path.

Being able to move in a circular path means it is possible to cut a large circular hole with a smaller diameter cutter. In theory, a very large hole could be cut with a very small cutter, although in practice this is not such a good idea, as we will see.

Fig. 6-2 shows the general technique. Assume for the moment that the centre of the circle is at (0, 0) and that on the inside of the circle there is a pilot hole large enough for the cutter to sit inside that hole, at position 1.

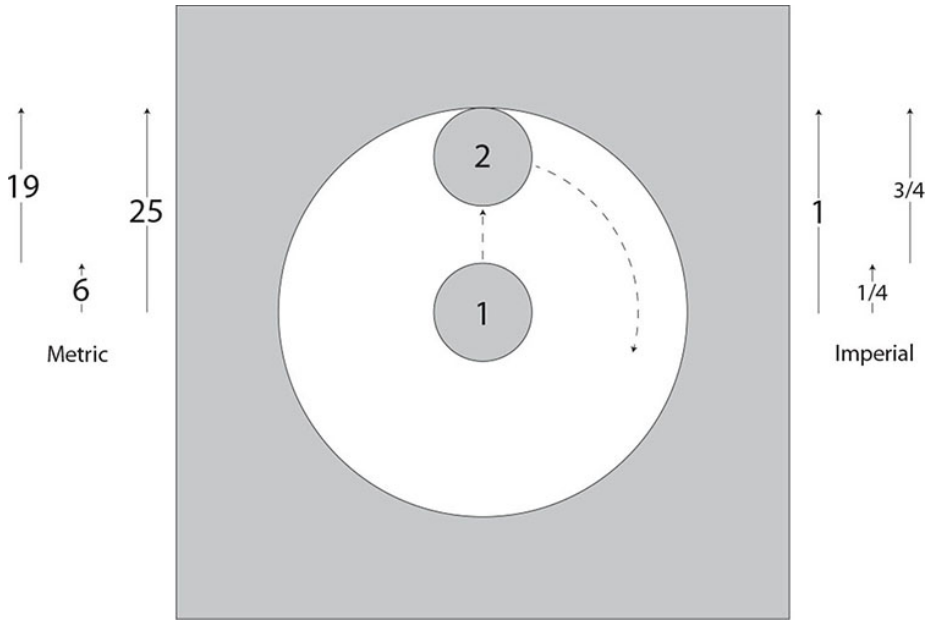


Fig. 6-2 A general method of cutting a large circular hole with a smaller diameter cutter.

Lower the cutter into the centre hole, then move it outwards so that the periphery of the cutter is touching the inside wall of the hole at position 2; then move the cutter in a complete circle.

When positioning the cutter inside the periphery of the circle, take account of the diameter of the cutter, so that for the 50mm (2in) circle shown in Fig. 6-2 and a 12mm (½in) diameter cutter such as an end mill, the method might be:

- Position the cutter above the centre of the pilot hole.
- Lower the cutter into the pilot hole, deep enough for the first cut.
- Cut out to make the periphery of the cutter touch the periphery of the circle.
- Move the Controlled Point in a circular path, clockwise (to take a conventional milling cut).
- If there are deeper cuts to be taken, cut downwards then repeat the circular cut.
- When the hole is to depth, retract to safe Z.

Table 6-1 Making a single cut

Instructions for millimeters	Alternative instructions for inches
If safe Z is 10mm	**If safe Z is 0.5in**
Circle radius is 25mm	Circle radius is 1in
Cutter radius is 6mm	Cutter radius is ¼in
Depth of cut is 1mm	Depth of cut is 0.04
G0 X0 Y0 Z10	**G0 X0 Y0 Z0.5**
G0 X0 Y0 Z-1	**G0 X0 Y0 Z-0.04**
G1 X0 Y19	**G1 X0 Y0.75**
G2 X0 Y19 I0 J-19 Z-1	**G2 X0 Y0.75 I0 J-0.75 Z-0.04**
G0 Z10	**G0 Z0.5**
G0 X0 Y0	**G0 X0 Y0**

Table 6-2 Making multiple cuts

Instructions for millimeters	Alternative instructions for inches
If safe Z is 10mm	**If safe Z is 0.5in**
Circle radius is 25mm	Circle radius is 1in
Cutter radius is 6mm	Cutter radius is ¼in
Depth of cut is 2mm	Depth of cut is 0.08
Total depth is 6mm	Total depth is 0.25in
G0 X0 Y0 Z10	**G0 X0 Y0 Z0.5**
G0 X0 Y19 Z0	**G0 X0 Y0.75 Z0**
(Spiral down to Z-2.)	(Spiral down to Z-0.08)
G2 X0 Y19 I0 J-19 Z-2	**G2 X0 Y0.75 I0 J-0.75 Z-0.08**
G2 X0 Y19 I0 J-19 Z-4	**G2 X0 Y0.75 I0 J-0.75 Z-0.16**
G2 X0 Y19 I0 J-19 Z-6	**G2 X0 Y0.75 I0 J-0.75 Z-0.25**
(Cut a flat bottom.)	(Cut a flat bottom.)
G2 X0 Y19 I0 J-19 Z-6	**G2 X0 Y0.75 I0 J-0.75 Z-0.25**
G0 X0 Y0 Z10	**G0 X0 Y0 Z0.5**

In practice, where the circle is to be cut to a greater depth and several cuts are required, it is better to spiral down then finish off with a flat bottom cut. That gives the cutter an easier entry to the hole and avoids having to cut repeatedly in a ‘flat’ circle then lower the cutter vertically into the work; or take it back to the centre, lower it and then move back to the periphery. Spiralling into the work also means there is no need for an initial hole to take the cutter.

If the hole is to be 6mm deep (¼in) and the depth of cut is 2mm (0.08in), the code might look like that in Table 6-2.

In Fig. 6-3, the solid line shows a rectangular shape. To cut around the outside of this shape with a 10mm (0.375in) cutter requires the Controlled Point to follow a path 5mm (0.1875in) away from each side, so that the edge of the cutter just touches the shape to be machined. At first glance, it might seem that the simplest way to cut this shape would be to use straight line movements as shown in Fig. 6-4, where the dashed line represents the path of the Controlled Point. The path of the Controlled Point will be spaced away from the rectangle edges by the cutter radius.

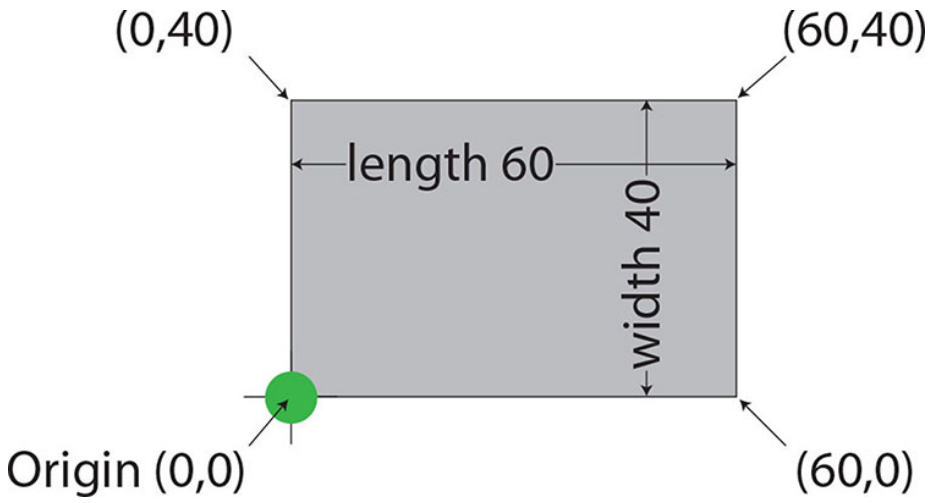


Fig. 6-3 A rectangular shape to be machined.

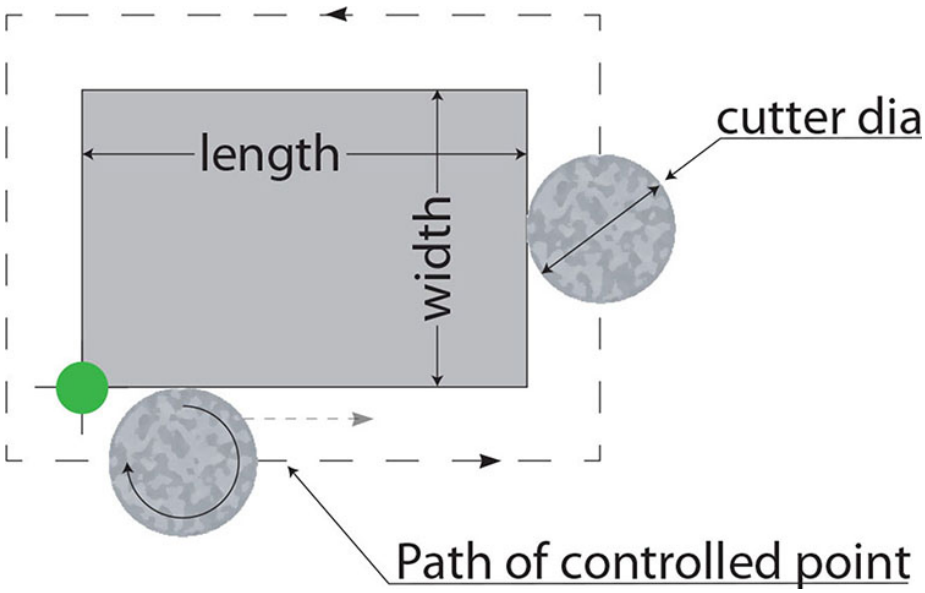


Fig. 6-4 The path of a cutter, using straight line movements.

However; this is not the best way to turn corners with a cutter, because at each corner the cutter must come to a stop then accelerate from zero speed in another direction.

Internally, most CNC software contains strategies for optimizing movement, which may mean the Controlled Point does not follow a given path exactly, but is allowed to deviate from that path to some extent (the extent of the deviation being defined in software or controlled by G code commands). Because of this, a better path is as shown in Fig. 6-5, where the Controlled Point moves along a path that pivots in an arc around each corner. Fig. 6-5 also shows that the path begins a short distance clear of the actual cutting path. This is the entry path, which helps the cutter to begin cutting the rectangle smoothly and helps the surface finish at the point where the cutter begins cutting the edge of the rectangle. There is more about entry paths later.

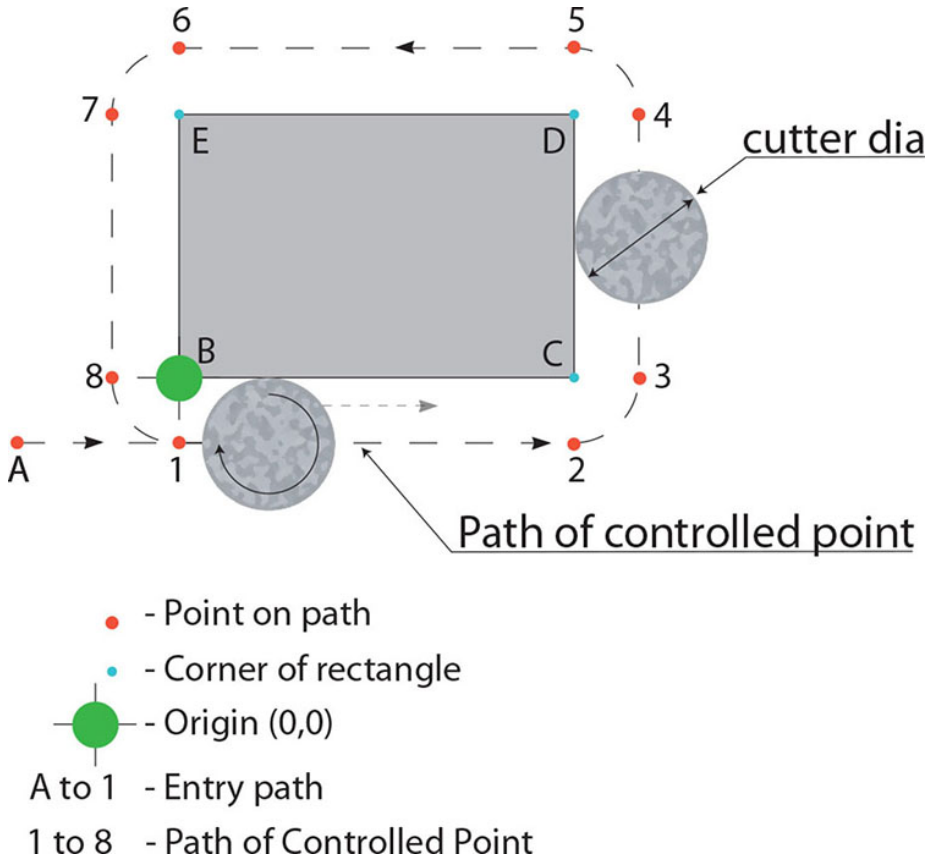


Fig. 6-5 A more effective path for a rectangular shape, using arcs at the corners

NAÏVE CAM

A Naïve Cam detector acts as a *trajectory planner*, examining the instructions coming up ahead in the program and attempting to optimize the overall movements of the Controlled Point to even out acceleration and deceleration, to stay close to the required path without having to make too many small movements and to manage the transitions from one line to another, from one arc to another or between lines and arcs.

The path of the Controlled Point can be programmed using a combination of straight line moves and circular arcs. In Fig. 6-5, the Controlled Point begins at point A and travels to point 1 then point 2.

From 2 to 3 it moves through a quarter circle. That pattern is repeated as it travels around the path shown, cutting the work to produce the rectangle as it goes.

The corner point C is (60, 0) and if the cutter radius is 6mm, point 2 will have coordinates (60, −6) and point 3 will have coordinates (66, 0).

Point 3 is the end of that arc, so if we use a **G3** command to get round that arc, we need I-6 J0 because:

- $X_{point\ C} – X_{point\ 2} = 60 – 60 = 0$
- $Y_{point\ C} – Y_{point\ 2} = 0 − (−6) = 6$

So the corner turning command is:
G3 X66 Y0 I0 J6

Assuming cutting starts from Z0 at point A(−20, −6) then ramps down to Z−1.5 at point 1 and continues at that level around the path, the complete set of commands to move around the path of the Controlled Point for this shape from point A to point 8 is as follows.

G0 X−20 Y−6 Z0
(Ramp down to point 1.)
G1 X0 Y−6 Z−1.5
(Straight path to point 2.)
G1 X60 Y−6
(Arc to point 3.)
G3 X66 Y0 I0 J6
(Straight line to point 4.)
G1 X66 Y40
(Arc to point 5.)
G3 X60 Y46 I−6 J0
(Straight line to point 6.)
G1 X0 Y46
(Arc to point 7.)
G3 X−6 Y40 I0 J−6
(Straight line to point 8.)

G1 X−6 Y0
(Arc to point 1.)
G3 X0 Y−6 I6 J0

Turning More than Once on the Way Around (LinuxCNC Only, from Version 2.5)

Fig. 6-2 showed one way of cutting a hole, and the associated G code showed how to spiral down, one turn at a time. However; there is a form of the **G2** and **G3** commands that allows one command to perform more than one revolution from the current point at the start of the arc to the end point of the arc. With care, this allows a deep hole to be cut by specifying the total depth of cut and the number of turns to be made as a single command. Just add a P~ value at the end of the G2 or G3 command.

If the hole is to be 50mm diameter, and the cutter is 10mm diameter, the centre is at 0,0 and the current coordinates of the Controlled Point are (20,0,0).

G2 X20 Y0 Z-5 I-20 J0 P10 will make the Controlled Point spiral down 5mm (to Z−5) spread over ten turns (P10), which is equivalent to typing ten individual commands to spiral down 0.5mm each revolution. This version of the command is capable of providing the basis for a thread milling sequence. For a flat-bottomed hole, make one extra cut at the bottom, at a constant Z height.

Creating a Circular Pocket (Mach3 Only)

The **G2** and **G3** commands machine an arc or a circle. Normally, there is a preliminary move out to the periphery of the arc or circle (as shown in Fig. 6-2). The **G12 I~** and **G13 I~** commands are similar to **G2** and **G3** commands, but can machine a circular pocket by incorporating the initial move to the periphery. If the current coordinates are at X, Y, the I~ parameter specifies the distance to move in the X direction, from the current point, to the periphery (for

example, **G12 I6** or **G13 I25**). The Controlled Point then moves in a complete circle, centred on the original coordinates X, Y, then returns to the centre. The cutter must be at the appropriate depth before using these commands, but they save having to calculate the offsets required by the **G2** and **G3** commands.

COMPLEX CURVES

CAD programs, some CAM programs, and vector drawing programs such as Adobe Illustrator are capable of creating compound curves of considerable complexity. In these programs, continuous lines used to draw objects typically consist of multiple segments and are often referred to as multi-segment lines or polylines. If these lines are created in a drawing program, they are called *paths* and each consists of a collection of straight or curved segments that contain anchor points, direction lines and direction points (or handles) as shown in Fig. 6-9. The anchor points define the positions of the associated points on the path, and the shape of the path on either side of an anchor point is controlled by the handles. Pulling the handles away from the anchor point will increase the section of path under the control of the handles, and rotating the handles will change the shape of the path on either side of the anchor point, so that the path is always at a tangent to the line. Visually, it is relatively easy to manipulate a path to produce a pleasing curve.

However, the consequence of all of this is that the mathematics involved in determining the start points, end points and radii of arcs within most polylines is an outstanding challenge even for those with a doctorate in geometry, so using manual methods to write G code programs for shapes that use complex curves is out of the question.

One way for CAM software to deal with paths, whether curved or not, would be to divide the path into straight line segments. If the original line is very curved, there might need to be lots of very short straight lines to come close to the shape of the original curved line, but the method would still work.

Project 6.1
Steady Bar and Foot

This steady bar is part of the mechanism that drives the minute hand on a clock. Instead of using gears, the clock uses a daisy wheel, which is a lobed cam wheel. To work, the wheel needs a steady rod that projects out from the wheel and sits between two pins on the front plate of the clock, just behind the dial. Fig. 6-6 shows part of this mechanism and Fig. 6-7 shows the dimensions of the steady pin that has a shaped foot. This is a small part and could be difficult to make by traditional methods, but it can be made as a solid piece using an end mill.

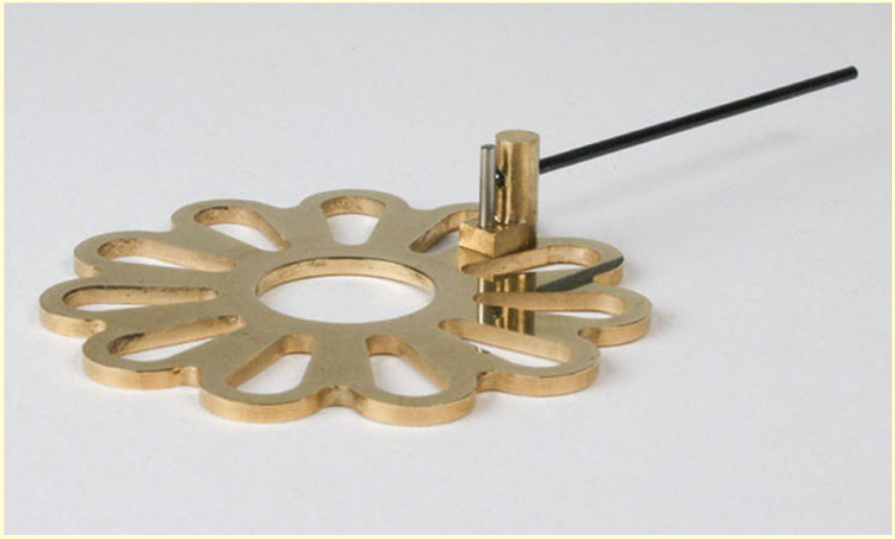


Fig. 6-6 Steady pin in position on a daisy wheel.

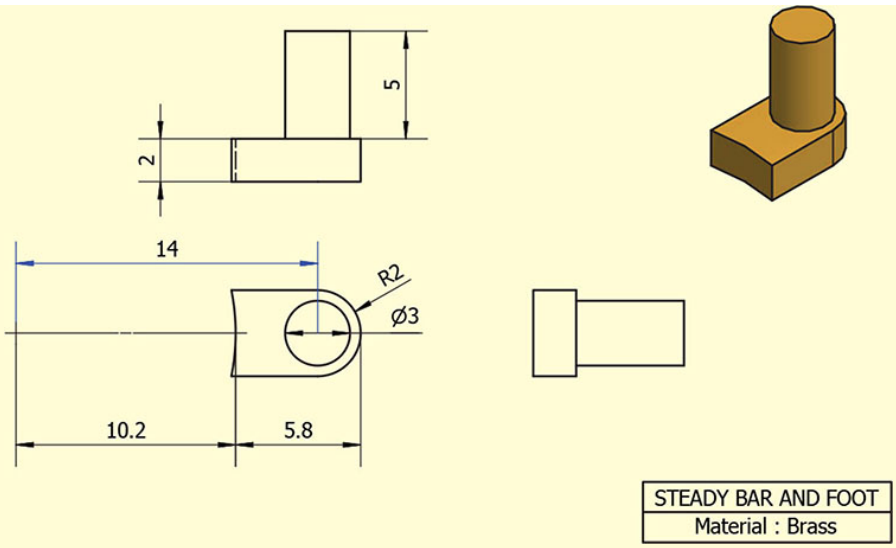


Fig. 6-7 Steady pin dimensions. The dimension shown in blue has been calculated from other dimensions.

Design Notes

Although the pin has a cross-drilled hole for the rod, a tapped hole to take a screw from the underside of the foot and a hole for a clock pin near the front of the foot, these details have all been omitted as they are not easily produced in the mill.

Material

- The steady is made from brass.

Tools

- A sharp end mill or slot drill at least 6mm (¼in) diameter.

Speeds and Feeds

- Run a four-tooth, 6mm end mill at 3,000rpm and use a feed rate of up to 200mm/min (8in/min). An end mill with fewer teeth or a larger diameter should be run at a proportionately lower speed.

Method

This is a small part and Fig. 6-8 shows one way of holding suitable material. The central axis of the chuck does not need to be aligned with the spindle and the chuck does not rotate, but this set-up provides a convenient way of gripping the bar. The part could also be machined from a piece of square or rectangular section bar held vertically in the vice, protruding above the jaws.

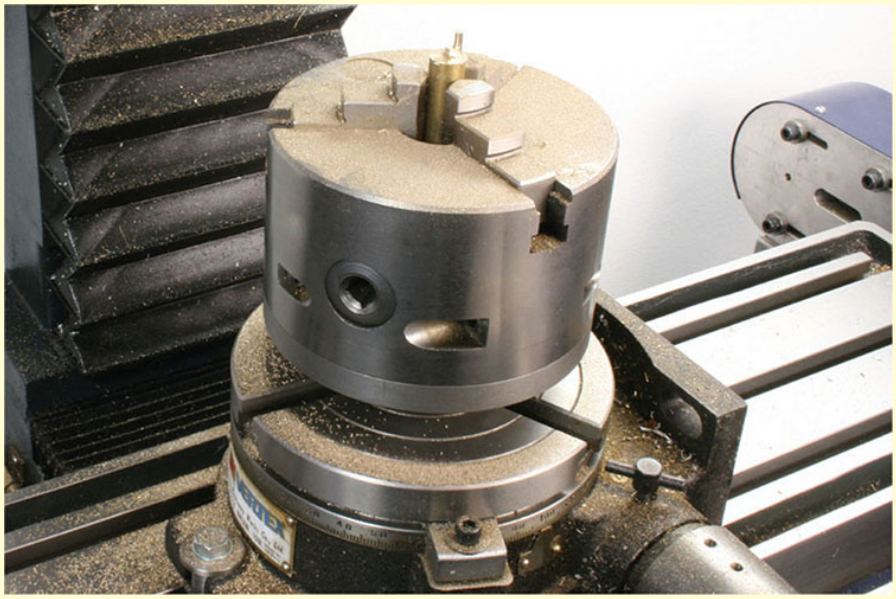


Fig. 6-8 One way of holding a bar to machine the steady pin from the end.

You might choose to make the work origin coincide with the centre of the vertical pin in the finished component. That way, the pin is machined as a circular profile 5mm deep. The foot has one semicircular end centred on the work origin, and the other is part of a circle centred 14mm to the left of the work origin. That dimension is shown in blue on Fig. 6-7 because it is a calculated dimension that is not required to specify the shape, but is useful when setting out the machining locations. It is found by adding the radius of the left-hand circular arc (10.2mm) and the distance from the right-hand edge of the foot to the nearest part of the left-hand

curve (5.8mm), then subtracting the distance from the right-hand end to the work origin (2mm). Without knowing the coordinates of the points where the circle cuts the straight sides, machine the curve as a complete circle centred at (−14, 0) with radius 10.2mm, or as a semicircle from the top to the bottom of the imaginary circle centred at (−14, 0). The top will be at (−14, 10.2) and the bottom will be at (−14, −10.2).

Although the flange is shown as 2mm deep, you might make it rather thicker, to include an allowance for parting off in the mill using a slitting saw, or in the lathe using a parting tool. Just take care it does not disappear as it separates from the parent material.

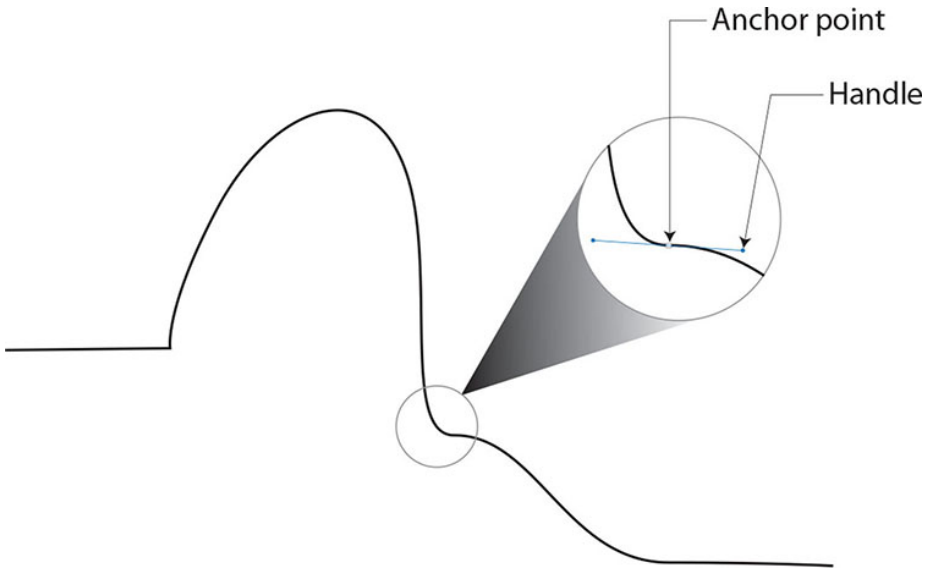


Fig. 6-9 An anchor point with handles.

Straight line sections will be reproduced as straight linear moves, while curved parts can be divided into short sections consisting of straight line moves. To give a more-or-less faithful reproduction of a curved line, the CAM program may use very short sections of line. One limitation of CAM software is the extent of the approximation that might be used to recreate a curve. When the output of the CAM program is used within a CNC program, there are user-defined

settings to control the extent to which the CNC program follows the shape given by the G code from the CAM program.

One way of looking at this is that the CNC program should always reproduce the shape exactly, but the problem with this is that with a very large number of very small movements, the program is likely to take a long time to complete its run. In a commercial machine shop, time is money, so this is important.

A different way of handling this is to allow the CNC program to look ahead to the instructions coming up next and to approximate the shape of the line being machined, so that the cutter follows the general shape of the path but need not pass through every point exactly. For a lot of shapes, particularly where a curve is for artistic effect rather than being a closely mating surface, this is good enough.

There is more to this, though, and the CNC program can be put into different modes to control how closely the program must follow the points on the defined path.

Exact path mode makes the Controlled Point visit each programmed point without deviation and ensures that the programmed path is reproduced exactly. Because each line segment involves accelerating from the beginning and decelerating towards the end, a path with a large number of points will take a comparatively long time to machine. It may also be difficult to ensure that an appropriate cutting speed is maintained throughout the program because of changes at the beginning and end of each segment, so surface finish may suffer and there may be visible changes in the surface finish along the path and within each segment.

Exact stop mode is like Exact path mode, but causes the Controlled Point to stop at each point. This is useful where there are frequent large changes in direction on a path, but it does slow the cutting speed considerably.

Best possible speed mode means the program will prioritize speed of movement over accuracy of the path and will attempt to maintain the set speed by sacrificing the accuracy with which the Controlled Point follows the path. In a variation of this, the extent to

which the Controlled Point may deviate from the defined path (that is, the tolerance) can be controlled by an additional parameter. This mode may cause sharp corners to be rounded.

Mach3 and LinuxCNC differ in the ways they implement these modes.

LinuxCNC:
G61 ***Exact path mode***
G61.1 ***Exact stop mode***
G64 ***Best possible speed mode. This mode attempts to maintain feed rate at the expense of accuracy in following the path specified for the Controlled Point.***
G64 P~ Q~ ***Best possible speed within the specified tolerance. The toolpath will stay within P~ of the specified path; and successive moves that are within Q~ of a straight line will be treated as straight lines. The naive cam detector will be active in this mode.***

Mach3:
G61 ***Exact stop mode***
G64 ***Constant velocity mode. In this mode, some aspects of the behaviour of the Controlled Point can be defined in the*** **Configure Logic** ***control panel.***

Some experimentation may be needed, initially, to work out the settings that best suit most jobs. Fig. 6-10 shows a rectangular pocket that has been machined with too great a tolerance on the extent to which the path of the Controlled Point may deviate from the specified points. The priority in this case was in favour of speed at the expense of the exact path (**G64**), but this has resulted in the actual path missing some areas, leaving four columns of material in what was supposed to be a cleared-out pocket. Reducing the

tolerance by reducing the P value in the **G64 P~** command resolved the problem while maintaining a decent speed.

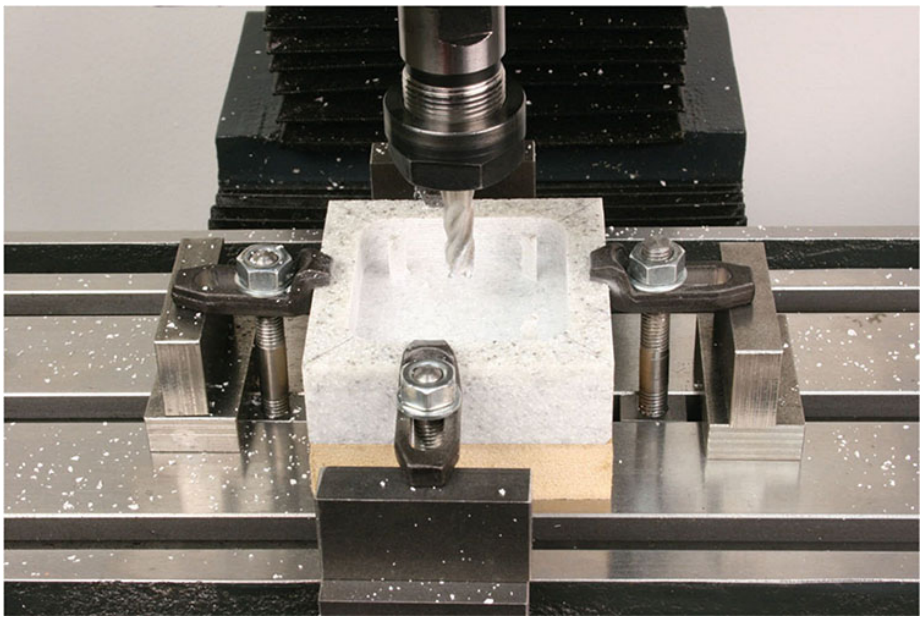


Fig. 6-10 Columns remaining in a recess due to too large a tolerance in the allowed deviation from the specified path.

So, the best way to deal with complex 2D shapes consisting of polylines is by drawing the shape in a CAD program, or a compatible vector drawing program, then employing CAM software that uses the mathematical definitions of the polylines to calculate the path for the Controlled Point. The output is G code suitable for the CNC program to drive the mill, but the way in which the CNC program follows the instructions will depend on the **G61/G64** commands.

Like the linear shapes in Chapter 5, shapes made with polylines will be profiles or pockets, or combinations of the two, and the same methods can be used to calculate the toolpaths.

For 2D shapes, Cut2D will output G code to allow complex shapes to be machined. Shapes can be drawn directly in Cut2D, so that it acts as the CAD program before taking information about the cutter and calculating the toolpath. For polylines, however, Cut2D is restricted to drawing straight line segments. Its big brothers, VCarve

Pro and Aspire, have a more extended set of drawing tools; although you may prefer to use a fully featured drawing program such as Adobe Illustrator, save the drawing and then import it into your CAM program.

Draw the shape shown in Fig. 6-11 directly within Cut2D. Define the material as 50 × 100mm, 1.6mm thick, with the X, Y origin at the bottom left corner.

Fig. 6-11 A shape that would benefit from tabs to hold it in position.

Use the polylines tool from the tool palette on the left side of the screen, adding the points in Table 6-3 to create the shape.

Actually, we can improve that shape by editing four of the points. Choose the Node editing tool and change points C, D, I and J by first selecting the outline, then right-clicking on a node and choosing Properties… Change those points to the ones in Table 6-4.

Table 6-3

Point	A	B	C	D	E	F	G	H	I	J	K
Coordinates	10,10	10,40	30,40	30,30	75,30	75,45	80,45	90,20	30,20	30,10	10,10

Table 6-4

Point	A	B	C	D	E	F	G	H	I	J	K
Coordinates	10,10	10,40	20,40	30,30	75,30	75,45	80,45	90,20	20,20	20,10	10,10

Select the resulting shape, and open the Toolpath panel by clicking the tab on the right of the screen. Define the material thickness as 1.6mm with the Z origin at the top surface.

Choose the Profile tool and a 3mm end mill to machine outside the profile of the shape. Calculate the toolpath and then preview that path. Do not machine it yet.

One of the benefits of a good CAM program is that it can easily cope with useful additions to the toolpath that would be difficult to program by hand. One of these additions is ‘tabs’ to assist with workholding.

Tabs

If you were to machine the shape shown in Fig. 6-11 from thin sheet, you would find there was a danger that as the final cut was completed, the finished workpiece might fly out of the sheet, or at least might move slightly and be damaged by the cutter. It is not possible to clamp this shape without the cutter running into the clamp at some point, so there is clearly a problem here. This is not an isolated instance because shapes like this all suffer from the same problem. Double-sided tape, strong glue and vacuum tables are all useful in this situation, but lead to their own problems. The tape may not be strong enough and its grip will be affected by dust on the workpiece. The glue may be strong enough, but may be difficult to release or to clean off the finished work. And vacuum tables require additional vacuum pumps, as well as being relatively expensive.

Fortunately, Cut2D is capable of dealing with this situation by creating small ‘tabs’ that continue to hold the cut shape in place until machining has been completed. These tabs can then be broken or cut quite easily to release the work. Although this does mean there may be a small amount of cleaning up to do, it is a good solution to this problem. Tabs can be created within the Profile toolpath panel. Experiment with placing four tabs around the edges of the shape. Once you are happy with the result, calculate the toolpath once

again, preview it and then return to the main Toolpath panel. Click on the Save Toolpath icon (the floppydisk symbol), and choose a postprocessor to suit your CNC software. This may be Mach3 Arcs (millimetres or inches) or EMC2 G61 (or LinuxCNC, which was formerly called EMC2).

From within your CNC program: Load that file, preview it in the Backplot window, set the work origin, position the cutter safely and run the program. The Controlled Point should follow the path shown in the drawing, but will be a short distance away from the finished sides of the shape because the program takes account of the diameter of the cutter so that the finished sides end up the right size.

Incorporating complex features into the design of a part emphasizes how the balance of skills has changed. Where once the skill was in marking work out by hand and eye and then machining to the marks, that changed, even for manual machining, to using a knowledge of simple geometry to be able to machine a more complex part, often by using the aids built into the display units for two- or three-axis DRO systems. Straight lines are simple enough, but it is the curves that introduce real complexity and challenge.

Now, the main skill is in designing and drawing a part. Once drawn (CAD), the geometry of the part can be calculated and G code programs generated from the design drawing (CAM), then parts can be machined using CNC. If we can master those three stages, complex parts are easily within our grasp and the world becomes our oyster.

Project 6.2

Yacht Servo Arm

Radio-controlled model yachts use a servo to pull or release a cord attached to the sail boom to pull the sail in or let it out, depending on the direction and strength of the wind. The total travel of the cord is considerable, so the servo needs to be

powerful and it needs a long arm. Plastic arms can be bought, but they are often a little shorter than required and a strong breeze really demands a strong arm.

The arm shown in Figs 6-12 and 6-13 is made from aluminium to the dimensions shown in Fig. 6-14. It is cut as a flat shape and then bent as required to fit the space under the deck. The arm is bolted or screwed on to a standard disc arm supplied with the servo.

Material

- Aluminium sheet 1.6mm ($^{1}/_{16}$in) thick.

Tools

- Centre-cutting end mill or slot drill 2mm ($^{5}/_{64}$in) diameter.

Speeds and Feeds

A cutter of this diameter can be run as fast as your mill spindle will turn, up to 10,000rpm. Theoretical feed rates can be up to 500mm/min (24in or 2ft/min).

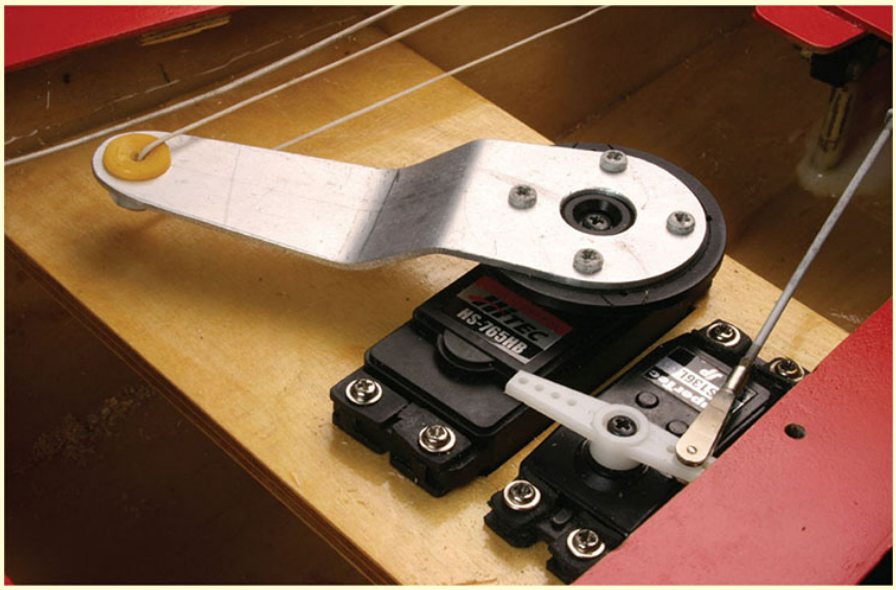


Fig. 6-12 A servo arm in a model yacht.

Because most mills cannot operate at those speeds, and small end mills are liable to snap under the cutting forces, realistic speeds and feeds are 3,000rpm and 50 to 100mm/min (2–4in/min).

Method

- Draw the shape, defining some tabs on the periphery.
- Clamp a sheet of material to a flat backing board (MDF or flat aluminium tooling plate).
- Machine the holes, then the profile.
- Release the machined part from the rest of the sheet by cutting through the tabs with a fine saw, and clean up the profile with a file.

Using a small cutter means all the holes can be milled and the profile cut using the same cutter. Using tabs means there is no need to add additional clamps or screws to hold the part before cutting the profile. All of this speeds the work.

Drawing the Shape

This is a deceptively simple shape that needs care at the drawing stage. The shape should be drawn as a polyline so that the profile is continuous, but the challenge is in drawing the arcs at the ends of the arm while ensuring they are tangent to the lines joining the arcs to form the sides of the profile.

Fig. 6-13 The flat shape of the servo arm.

One way to draw the shape is to create it directly in Cut2D or VCarve Pro. However, the limited tools in Cut2D make this difficult to do accurately because:

- polylines are limited to straight lines in this program;
- it is difficult to identify the tangent points between the circular arcs and the straight lines, especially as there are no tools for drawing arcs.

One way of dealing with this is to consider how accurately this shape needs to be drawn. The centres of the holes need to be positioned accurately and that can be done without any trouble. The outline does not need to be any more than visually pleasing. Begin the outline by drawing two circles centred on the locations of

the two largest inner circles. Draw a polyline as shown in blue in Fig. 6-15. Select the two circles and the polyline, and use the Weld Selected Vectors tool to make one composite shape. If the curves seem to flow reasonably smoothly into the straight sides, leave it at that. If it looks a bit ‘bumpy’, Undo and then edit the nodes before trying again.

VCarve Pro has better drawing tools and does have a tool for drawing arcs, but it is still difficult to identify tangent points so the same method can be used as for Cut2D.

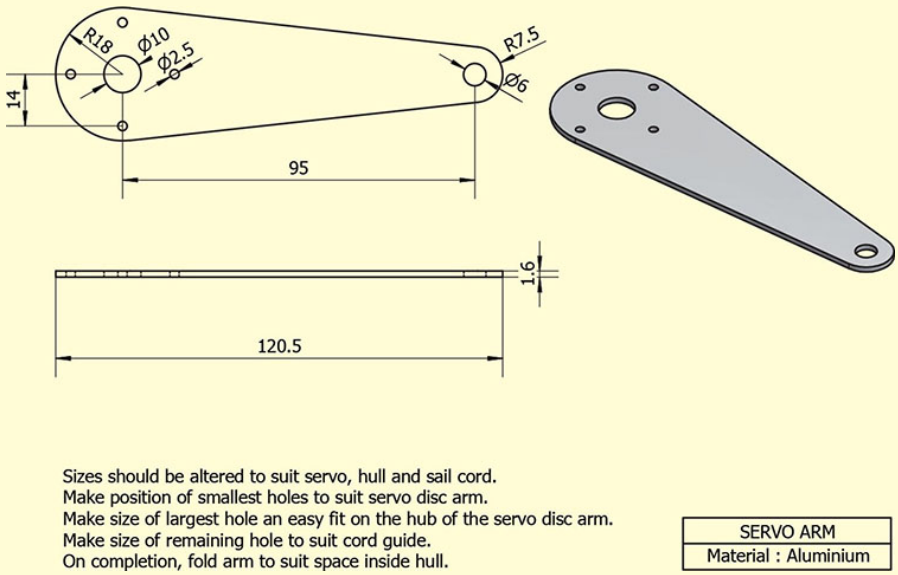


Fig. 6-14 Dimensions of the servo arm.

You may find it is easier to do this drawing in your favourite vector drawing program, although even the very best of these may suffer from the same difficulty in identifying tangent points and so might not offer much advantage for this kind of shape. Use lines as guides until you get a good fit between lines and circles.

It is likely that the best way to draw this shape is in a proper CAD program such as AutoCAD, using construction lines to draw two circles and the tangent lines that form the straight sides. Use tangent constraints between those lines and circles, but watch out for the shape moving out of alignment as you create those constraints. Take a deep breath and go carefully.

Switch to ‘normal’ lines and then use the construction lines as guides to draw a single polyline consisting of the arcs and the straight lines, making use of the tangent points created by the construction lines and circles as end points for the arcs and lines.

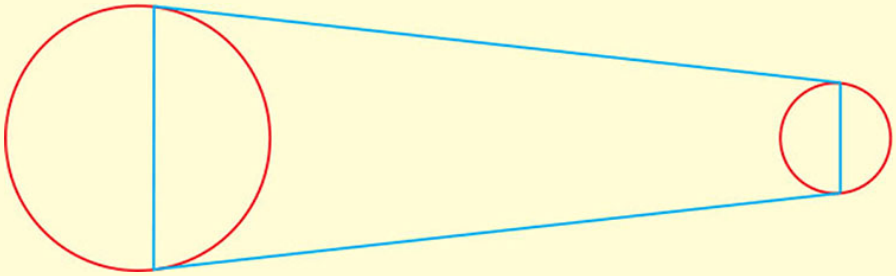


Fig. 6-15 Two circles and a polyline to help draw the outline of the servo arm.

Add the holes last. Make the spacing of the small holes to suit the existing holes in the disc arm supplied with your servo. Save the drawing as an EPS or DXF file and then import it into Cut2D or VCarve Pro.

Creating the Toolpaths

Once the drawing has been completed and is in Cut2D or VCarve Pro, define your toolpaths, adding tabs to the profile, and save the toolpaths as one file in a form suitable for your CNC software. The order in which you arrange the toolpaths is important, as that will be the machining order, so put the toolpaths for the holes at the top of the list and the toolpath for the profile last, because that contains the retaining tabs.

Treat the holes as pockets and not drill toolpaths, and be kind to your cutter by ramping rather than plunging into the work..

The tops of these parts were produced using a fourth axis to turn each workpiece to allow another face to be machined.

7 Subroutines, Loops and Decisions

In this chapter you will learn how to:

- create and use subroutines;
- create and use loops;
- take decisions (LinuxCNC only);
- incorporate G code created by a CAM program into a subroutine.

SUBROUTINES

One of the useful things about a CNC machine is that it can undertake repetitive tasks very easily. That means it can remove a lot of the drudgery from many jobs in the workshop.

Within a CNC program, a subroutine is a section of the program that can be used to hold the instructions for a task. Whenever that task has to be carried out, the program can call on the subroutine to carry out those instructions. Every time the subroutine is called, the same instructions are carried out, so you could type instructions into a subroutine just once and have the CNC program use those instructions hundreds or thousands of times. Subroutines are useful for performing a repetitive task while machining a single workpiece, but they are also useful for machining multiple workpieces at one go.

You can avoid the use of subroutines entirely by using a CAM program, but writing your own program, with your own subroutines, does give a great deal of flexibility in many cases. You can also use subroutines to hold sections of code that have been created in a

CAM program, giving you a great deal of control but avoiding any difficult calculations.

Inside a program, subroutines are separate blocks of program instructions, and they need a specific first instruction and a specific end instruction to distinguish them from the rest of the program.

O codes are used to define subroutines and each subroutine must have a reference number. Note that O codes use the letter O, not the digit zero. In this printed book, the letter O and the digit 0 have different shapes.

LinuxCNC: In LinuxCNC, subroutines must be placed before the start of the main program block. The start of a subroutine consists of:
the letter O
followed by a reference number (any integer between 0 and 99999)
followed by the word sub
A typical start to a subroutine might be: *O120 sub*
where O denotes the start of a subroutine, and 120 is a reference number used when we need to refer to this subroutine later. The word sub denotes the start of a subroutine rather than a block of code used for another purpose (for example looping, repeating or conditional branching).
The end of a subroutine is signalled by the O code, the subroutine reference number and then the word endsub, for example:
O120 endsub
To use the subroutine from within the main program, use O followed by the subroutine reference number and the word 'call' for example: O120 call

Mach3: In Mach3, subroutines must be positioned after the end of the main program block (that is, after the last* M30 *or* M2 *instruction).
The start of a subroutine consists of: the letter O followed by a reference number (any integer between 0 and 99999)
A typical start to a subroutine might be: *O120*
where O denotes the start of a subroutine, and 120 is a reference number used when we need to refer to this subroutine later.
The end of a subroutine is signalled by: *M99*
To use the subroutine from within the main program, use* M98 *followed by the letter* P *and the reference number of the subroutine:
M98 P120

Fig. 7-1 shows a design for a repeated pattern of holes. If a workpiece is mounted on a rotary table controlled using the A axis, the holes can be produced by repeating this sequence four times:

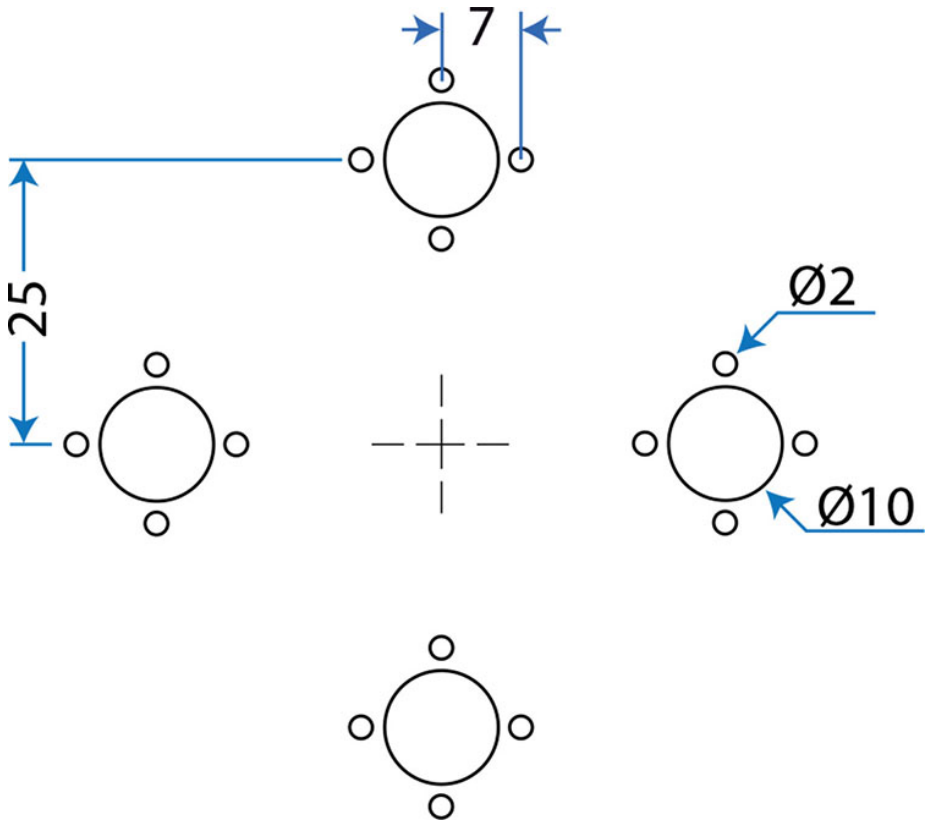


Fig. 7-1 A repeated pattern of holes.

- Machine one set of five holes.
- Turn the rotary axis through 90 degrees.

Here is a subroutine that can be used to cut a single group consisting of a 10mm diameter hole and four smaller holes using a 2mm centre-cutting end mill (Fig. 7-2) in aluminium sheet 1.6mm thick. The origin is set as shown in Fig. 7-3 and safe Z is set to 25mm (1in).

Fig. 7-2 A 2mm centre-cutting end mill.

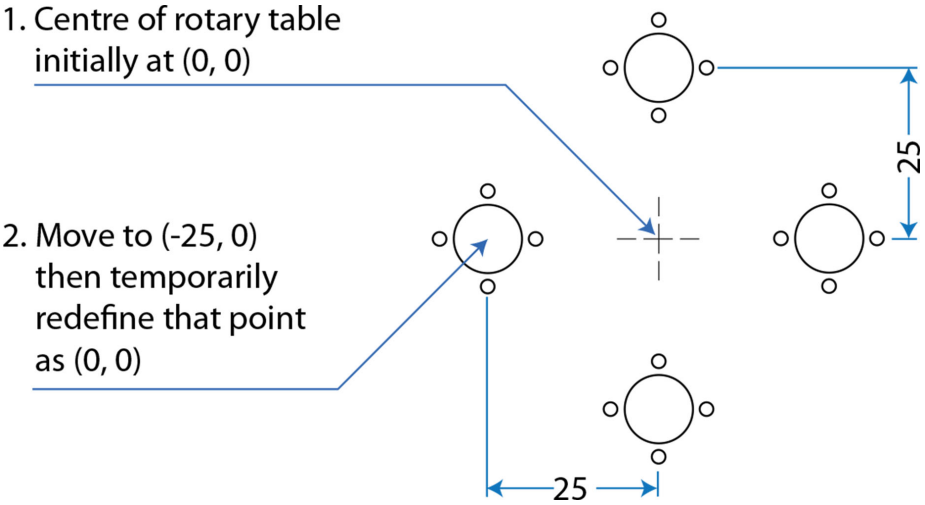


Fig. 7-3 Setting the origin for the work.

Note that the origin can be set by locating the centre of the rotary table and making that X25 Y0 by touching off the X axis with a **G54** offset of 25 then touching off the Y axis with a **G54** offset of 0. That has the effect of placing the origin X0 Y0 at the centre of the circular pattern to the left of the centre of the rotary table. A ~ specifies the

angle (in degrees) to turn the fourth (rotary) axis. Here is the code for machining one set of five holes.

```
(Machine the 10mm diameter hole)
G0   X0   Y4
G0   Z0
(ramping into the material)
G2   X0   Y4   10   J-4   Z-0.4
G2   X0   Y4   10   J-4   Z-0.8
G2   X0   Y4   10   J-4   Z-1.2
G2   X0   Y4   10   J-4   Z-1.6
(cut right through the material)
G2   X0   Y4   10   J-4   Z-1.8
(and one more at that depth of cut)
(because the last cut was a spiral)
G2   X0   Y4   10   J-4   Z-1.8
G0   Z1
(Drill the 4 × 2mm holes)
(by plunging down with the end mill)
G0   X0   Y7
G1   Z-2.5
G0   Z1
G0   X7   Y0
G1 Z-2.5
G0   Z1
G0   X0   Y-7
G1   Z-2.5
G0   Z1
G0   X-7   Y0
G1   Z-2.5
G0   Z1
```

The instructions above can be inserted into the programs below, as shown by the arrows (←). Each program calls the subroutine four times, producing four sets of five holes.

LinuxCNC

(Program start)
<<initialization block>>
F100
S3000 ***(depends on material being cut)***
0100 sub
← ***insert the code to machine one set of five holes***
0100 endsub
(Main program block)
G0 Z25
0100 call ***(call subroutine 100)***
G0 A90 ***(rotate the workpiece)***
0100 call ***(call the subroutine again)***
G0 A180
0100 call
G0 A270
0100 call
G0 A360
M30

Mach3
(Program start)
<<initialization block>>
F100
S3000 ***(depends on material being cut)***
(Main program block)
G0 Z25
M98 P100 ***(call subroutine 100)***
G0 A90 ***(rotate the workpiece)***
M98 P100 ***(call the subroutine again)***
G0 A180
M98 P100
G0 A270
M98 P100
G0 A360

```
M30
0100    (Subroutine to machine five holes)
←       insert the code to machine one set of five holes
M99
```

INITIALIZATION BLOCK

The initialization block, introduced in Chapter 5, contains the following commands.

Table 7-1

Instructions for millimetres	Alternative instructions for inches
(Start of initialization block)	
G21 (set units to millimetres)	**G20** (set units to inches)
G90 (use absolute distances)	
G94 (feed per minute mode)	
G92.1 (cancel offsets)	
G91.1 (incremental arc mode)	
G54 (use coordinate system 1)	
G98 (set retract behaviour for canned cycles)	
G49 (cancel tool length offset)	
G40 (cancel cutter compensation)	
G17 (select XY plane)	
G80 (cancel canned cycle motion mode)	
(End of initialization block)	

G92 AND G52 OFFSETS

G92 applies an overall offset to the current coordinate system to give the current point the new coordinates stated in the command. So **G92 X2 Y3 Z6** will work out and apply the offsets necessary to make the current point (2, 3, 6). This is useful, but needs to be treated with care as this is an additional offset applied to the current coordinate system and means more than one offset may be in operation, which can become confusing unless handled with care.

G92 is available within both Mach3 and LinuxCNC. The **G52** command is only available in Mach3.

G52 applies the offsets given after the command. So **G52 X2 Y3 Z6** applies offsets of 2, 3 and 6 to the X, Y and Z axes, respectively. This differs from **G92**, where the software calculates the offsets.

G52 offsets the current point by a given distance and, like G92, is an overall offset applied to the current coordinate system. It requires the same careful management as **G92**. In Mach3, **G92** and **G52** use a common offset system and should not be used together.

Temporarily Making the Current Point the Origin

G92 X~ Y~ Z~ A~ B~ C~
temporarily makes the current point have the coordinates that follow the G92. It does this by temporarily shifting the whole coordinate system.

The most useful application of this is to make the current point the origin, because this means subroutines can be written on the assumption that the machine is at the origin at the start of the subroutine. In the main program, moving to a position then temporarily making this the origin allows the subroutine to be used. Afterwards, the effects of the **G92** command can be cancelled by using **G92.1** or **G92.2**, and the rest of the program can continue, using the previous coordinates.

As an example of how useful this is, a set of die-cast boxes required three sockets, as shown in Fig. 7-4, to the dimensions shown in Fig. 7-5.

Each socket consists of three circular holes, but the difficulty is that the numbers used to specify those holes depend on where the centre of each circle lies. It is easy enough to write a program that has commands for each of the three separate socket holes, at each of the positions that require a socket, but this would be tedious if there were more socket positions.

CANCELLING G92 OFFSETS

The **G92** command stores its offsets in parameters 5211 to 5216.

G92.1 cancels **G92** offsets and puts zeroes into 5211 to 5216.

G92.2 cancels **G92** offsets but leaves the values in 5211 to 5216.

G92.3 is like **G92** but takes its offsets from parameters 5211 to 5216 instead of from values given after **G92**. This means that values for the G92 offsets can be passed from one program to another:

- Use **G92 X~ Y~ Z~ A~ B~ C~** in one program.
- If they have to be cancelled, use **G92.2** to leave the values in 5211 to 5216.
- In the next program, use **G92.3** to reload those values from 5211 to 5216.

Fig. 7-4 Holes for three XLR sockets to be cut in a box.

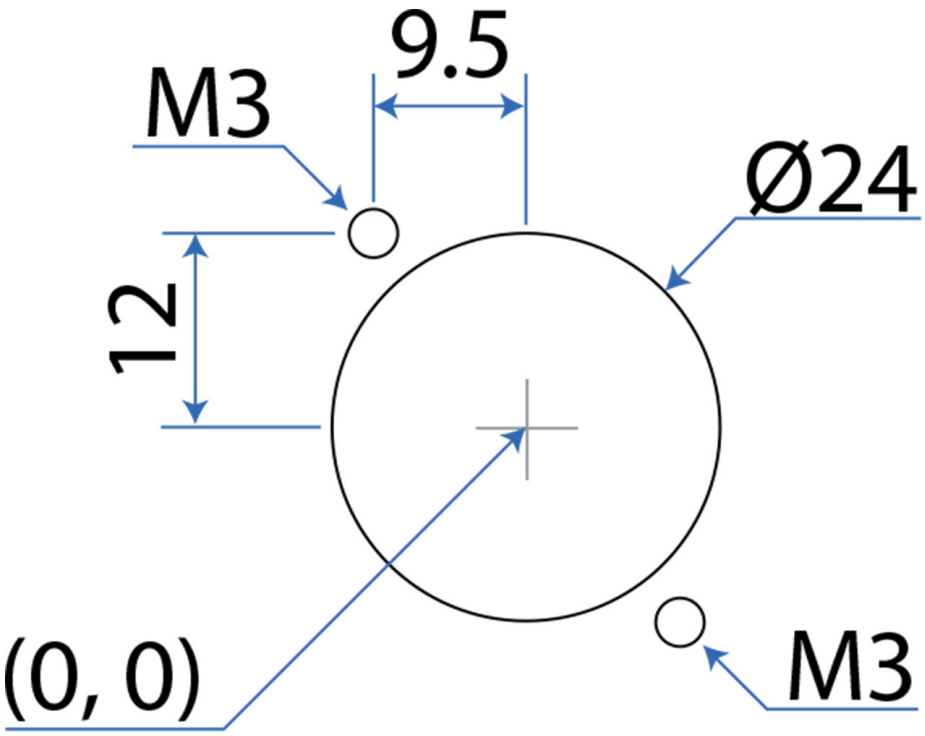


Fig. 7-5 Dimensions for a single XLR socket hole.

Creating a subroutine to machine a socket centred at (0, 0), then moving to each socket position and temporarily making that (0, 0),

means that subroutine can be called at each position and it will produce a socket there.

The following program cuts holes for three sockets, as shown in Fig. 7-6, with centres at (26.5,30), (26.5,60) and (26.5,90), in a die-cast box 3mm (⅛in) thick, using a 2.5mm diameter end mill running at 2,000rpm. The size of the end mill was determined by the fact that tapping size for **M3** is 2.5mm diameter, so that those two smaller holes can simply be drilled to finished tapping size in one vertical plunging operation. It would be easy to modify the procedure to use a 2mm diameter end mill (or any diameter smaller than 2.5mm) and produce the holes by cutting a 2.5mm circle using a **G2** command.

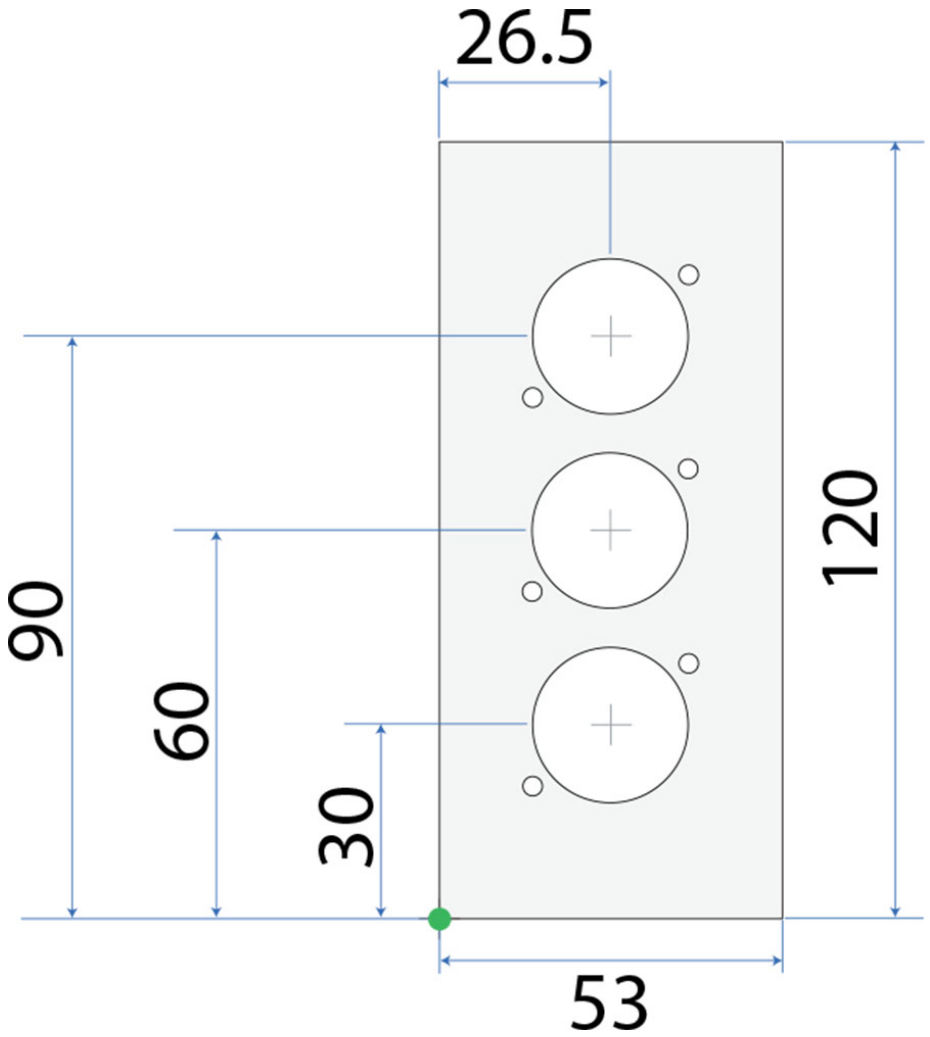


Fig. 7-6 Dimensions for a set of three XLR socket holes.

The origin for the whole job is set at the left front edge of the side of the box. Die-cast boxes have a significant taper on the sides, but sizes have been chosen so that by setting the origin at this corner, the taper can be ignored.

LinuxCNC:
(Program start)
<<initialization block>>
F100
S1000 ***(depends on material being cut)***
O100 sub ***(subroutine to machine a socket)***
(redefine the current point as 0,0)
(redefine the current point as 0,0)
← ***insert code to machine the holes for a socket (see below)***
G0 X0 Y0
G0 Z25 ***(Safe Z)***
(then restore the original coordinates)
G92.1
0100 endsub
(Main program block)
G0 Z25
G0 X0 Y0
(Go to the first socket position)
G0 X26.5 Y30
(then call the subroutine to machine the socket there)
O100 call ***(call subroutine 100)***
(Now do the next one in the same way)
G0 X26.5 Y60
O100 call
(and the third socket)
G0 X26.5 Y90
O100 call
G0 Z25
G0 X0 Y0
M30

```
Mach3
(Program start)
<<initialization block>>
F100
S1000 (depends on material being cut)
(Main program block)
G0 Z25 (Safe Z)
G0 X0 Y0
(Go to the first socket position)
G0 X26.5 Y90
(then call the subroutine to machine the socket there)
M98 P100 (call subroutine 100)
(Now do the next one in the same way)
G0 x26.5 Y60
M98 P100
(and the third socket)
G0 X26.5 Y90
M98 P100
G0 Z25
G0 X0 Y0
M30
O100 (Subroutine to machine a socket)
(redefine the current point as 0,0)
G92 X0 Y0
← insert code to machine the holes for a socket (see below)
(then restore the original coordinates)
G92.1
M99
```

Here is the code for the subroutine to machine the socket (but note that, compared to the standard upright view in Fig. 7-5, this socket has been rotated 90 degrees to suit the orientation of the box during machining – *see* Fig. 7-7).

(Machine the large hole first)
(It is currently centred at X0 Y0)
(To machine a single socket)

```
G0  X11        Y0      Z0
G1  X11        Y0      Z-1  I-11      J0
G1  X11        Y0      Z-2  I-11      J0
G1  X11        Y0      Z-3  I-11      J0
G1  X11        Y0      Z-4  I-11      J0
G0  Z                                 1
G0  X0         Y0
```

(Machine two 2.5mm tapping-size holes)

```
G0  X-12.25    Y-9.5   Z0
G1  X-12.25    Y-9.5   Z-1  10.25     J0
G1  X-12.25    Y-9.5   Z-2  I0.25     J0
G1  X-12.25    Y-9.5   Z-3  10.25     J0
G1  X-12.25    Y-9.5   Z-4  10.25     J0
G0  Z1
G0  X12.25     Y9.5    Z0
G1  X12.25     Y9.5    Z-1  1-0.25    J0
G1  X12.25     Y9.5    Z-2  I-0.25    J0
G1  X12.25     Y9.5    Z-3  I-0.25    J0
G1  X12.25     Y9.5    Z-4  I-0.25    J0
G0  Z1
G0  X0         Y0
```

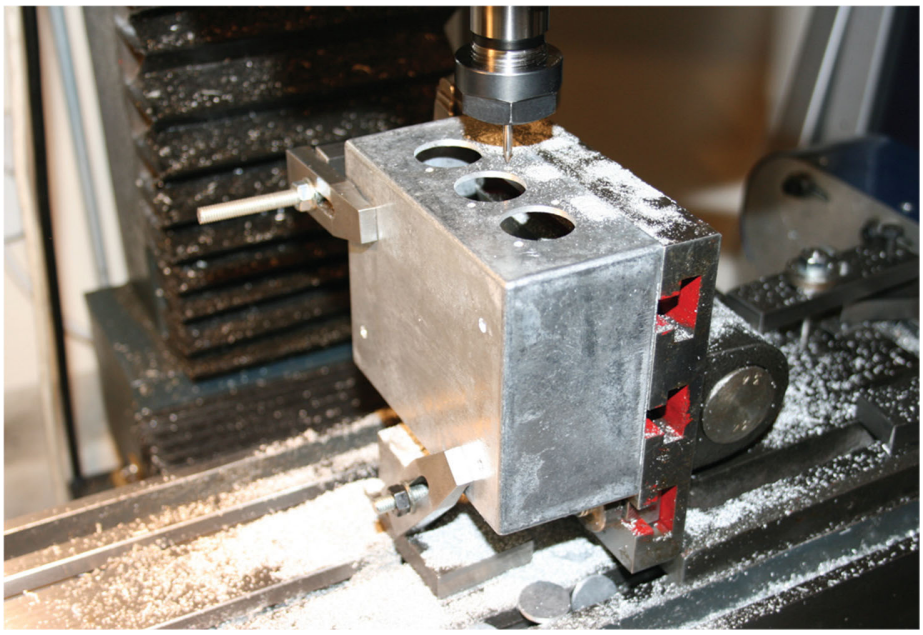


Fig. 7-7 Die-cast box held on an adjustable angle plate with the top surface horizontal.

All four sides of the box are tapered, so when machining the holes with a small-diameter cutter, it is important that the cutter 'sees' a flat side so that it takes the same depth of cut as it moves around, particularly at the beginning of the cut. To achieve this, the box was clamped to a tilting angle plate and arranged so that the top face was horizontal (Fig. 7-7). That helped preserve the life of the little cutter. Note that because the table was more conveniently secured across the table, the program cuts the sockets in line fore and aft, although the completed box will sit with the sockets running left to right (Fig. 7-4).

USING COMPOSITE BOARD

MDF and other composite boards are ideal for fixtures that use locating pins, and where machining forces are low. They are not so good where work is to be secured using woodscrews or machine screws, or where cutting forces are larger. Avoid securing workpieces by woodscrews alone; they will move.

LUBRICATING AND COOLING THE CUTTER

Tool life can often be extended, and the quality of a machined surface improved, by using appropriate coolant and lubricant. such as soluble oil or neat cutting oil.

Choose the coolant to suit the characteristics of the material being machined. Soluble oil is a mixture of water and a specially designed oil, and is an effective coolant for high speed steel tools machining steel.

Neat cutting oil is usually thicker than soluble oil, and it acts both as a coolant and a lubricant. It can be applied in the same way.

Other coolants include white spirit or water, for aluminium; and some specially formulated liquids designed to resist the pressures of machining hard materials.

Protect yourself against coolant being sprayed, and avoid inhalation of vapours from coolants. Contact and inhalation are detrimental to health, so you should wear protective clothing and appropriate breathing apparatus.

Carbide tools are designed to cut dry, as are some other tools which carry a metallurgical coating.

Some of these tools may tolerate flood coolant, but can be damaged by an intermittent or trickle supply of coolant.

Project 7.1

Sacrificial Plates

One common set-up is to put a flat sacrificial plate or board (Fig. 7-8) under a flat workpiece and clamp both to the mill table. Then you can run a cutter around any of the edges of the workpiece without worrying about whether the cutter will mark the table. It is useful to have a handy source of plates; otherwise there is a tendency to make do with something less suitable. When you

have a spare moment, it is a good idea to make a pile of plates for use later.

When using the plates, it is a nuisance having to cope with the plate moving around under the workpiece while you try to align the work and clamp it in position, so the ideal plate will have recessed holes to take bolts that can be used to secure the plate to the table. Then you only need to deal with aligning the workpiece.

Fig. 7-9 suggests sizes that can be changed to suit your own mill table. Note that the board in Fig. 7-8 has a bolt recess in the centre, rather than the 6mm hole suggested in Fig. 7-9. Choose the hole type to suit your own purposes.

Fig. 7-8 A sacrificial plate.

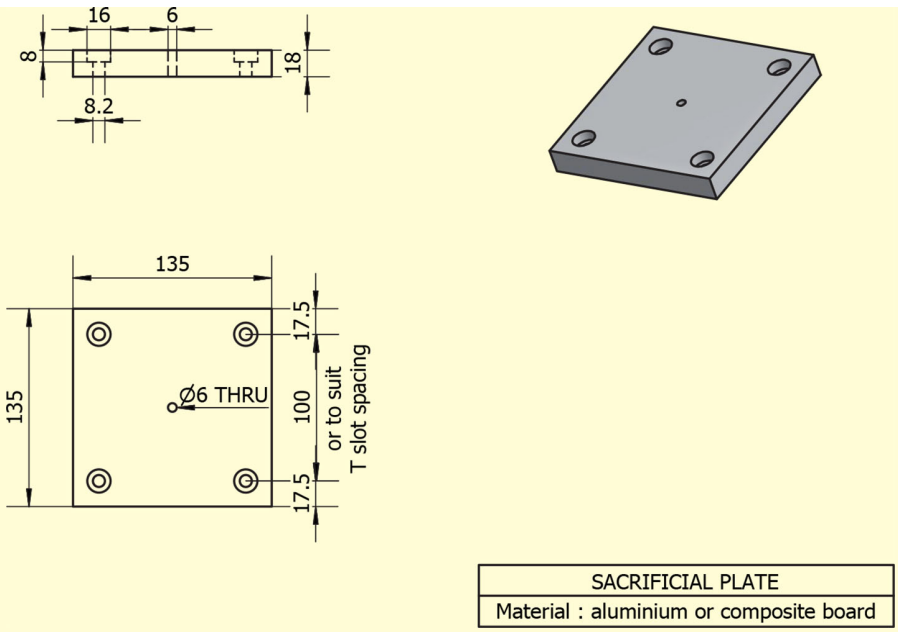


Fig. 7-9 Suggested sizes for a sacrificial plate.

A central hole to suit a standard diameter pin provides a simple way of locating workpieces. It is an easy way of making the plate into a fixture, at least until its top surface has been machined away. These are sacrificial plates and ideal for short-term low-run repetition work at low accuracy, but longer-term use deserves a dedicated fixture plate made of aluminium or steel.

You could also drill the plate for a set of pins that would act as guides to locate workpieces for repetition machining. These plates provide lots of opportunities for speeding up workholding.

Design Notes

The recess at each bolt hole is large enough to accommodate a standard socket to allow the bolts to be tightened. The depth of the recess allows a standard bolt head, with a standard washer, to sit under the level of the top of the board. On a large plate or a board, additional bolt holes may be required near the middle.

Material

- Any plate or board with uniform thickness, thick enough to take a recessed bolt head.
- Aluminium tooling plate, ground flat on both sides, makes a deluxe version, but 18mm (¾in) MDF works well if you take precautions to avoid the dust from machining.

Tool

- A 6mm (¼in) centre-cutting end mill or a wood routing bit, depending on the material.
- Use a cutter long enough to be able to reach the bottom of the hole (at least 18mm, if that is the thickness of plate you are using).
- High-speed steel tools will not last long if you are machining abrasive material such as a composite board, so use a carbide cutter if possible.

This size of cutter will work nicely in this size of hole and recess, because the width of the cutter allows it to create a flat bottom face in the recess in one rotation, as the cutter is wider than the ledge that forms that face (Fig. 7-10).

Speeds and Feeds

Run the cutter as fast as you can, especially if it is a router cutter; 5,000rpm is not unreasonable, but you will still get usable results at your mill's top speed.

Using 3,000rpm and a 6mm (¼in) two-tooth carbide cutter, the feed rate can be as high as 750mm/min (2.25ft/min) on an industrial machine. On a much smaller and less robust machine, 100–200mm/min (4–8in/min) is a more realistic feed rate to begin with. Aluminium and composite board are relatively soft, and the finish on both benefits from spindle speeds and feed rates as close to optimum as possible. Finishing cuts also benefit from

climb milling. With composite board, slow feed rates combined with high spindle speeds may cause burn marks on the board and overheating of the cutter.

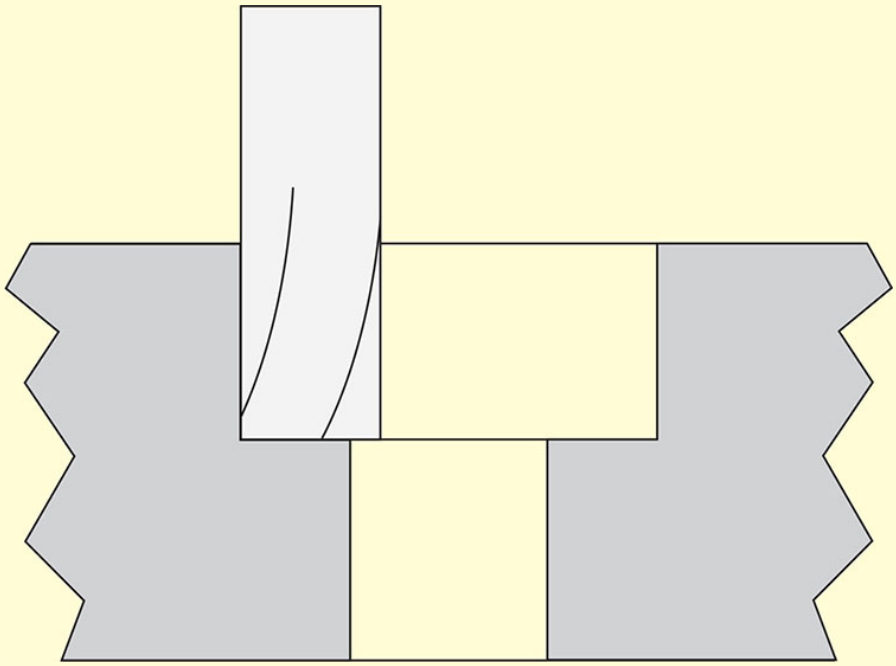


Fig. 7-10 Machining the stepped recess with a cutter large enough to cut the bottom step in one pass.

Method

For each hole, plunge drill the centre then open it out to the smaller diameter. Then machine a pocket to create the larger-diameter recess for the bolt head.

Creating a Program Manually

Because the same machining sequence is used at each hole location, it makes sense to create a subroutine that does those operations but assumes the centre of the hole is at (0, 0).

In the main program, create instructions for this sequence, for each hole:

- Move to the centre of the hole.
- Temporarily redefine that as (0, 0).
- Call the subroutine to machine the hole.
- Turn off the temporary coordinates.

Using a CAM Program

Draw the holes in a CAM program like Cut2D, using two correctly sized circles at each location. Then create one toolpath for all of the inner holes (which you should treat as pocket toolpaths) and a second toolpath for all of the bolt head recesses (which you should also treat as pockets).

Once you have created the two toolpaths, arrange them in order (inner holes then outer recesses) then save them as one program (by opting to save all toolpaths). Load the resulting program into your CNC software package, and machine the holes.

You will want to make boards and plates in a selection of sizes to cater for different sizes of workpieces, but the method is the same, regardless of the sizes.

Nesting Subroutines

The box shown in Fig. 7-11 contains rows of sockets and is one of a series of boxes in which the lids require rows of holes for similar sockets. Not all the rows are full of holes, though, and some have a mixture of full rows and incomplete rows, while others have rows with no sockets at all. Fig. 7-12 shows a lid with fewer rows and Fig. 7-13 shows the layout of the holes. Note that the origin is at the top left for this job.

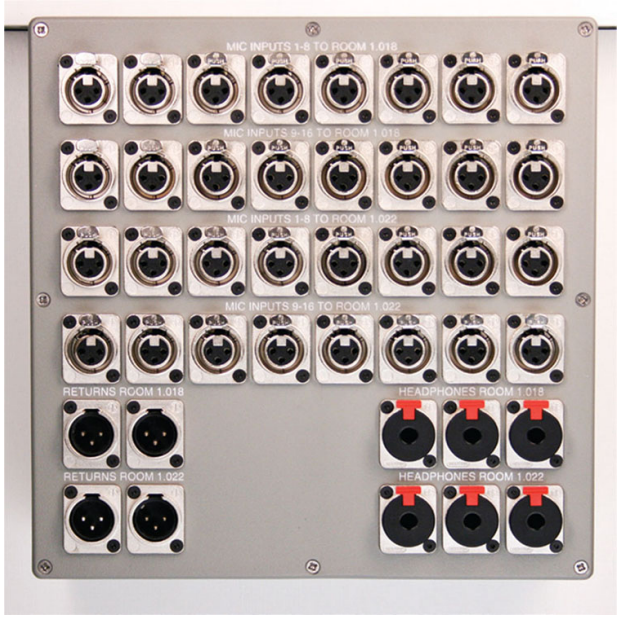


Fig. 7-11 A box with rows of XLR sockets.

Fig. 7-12 A box with incomplete rows of XLR sockets.

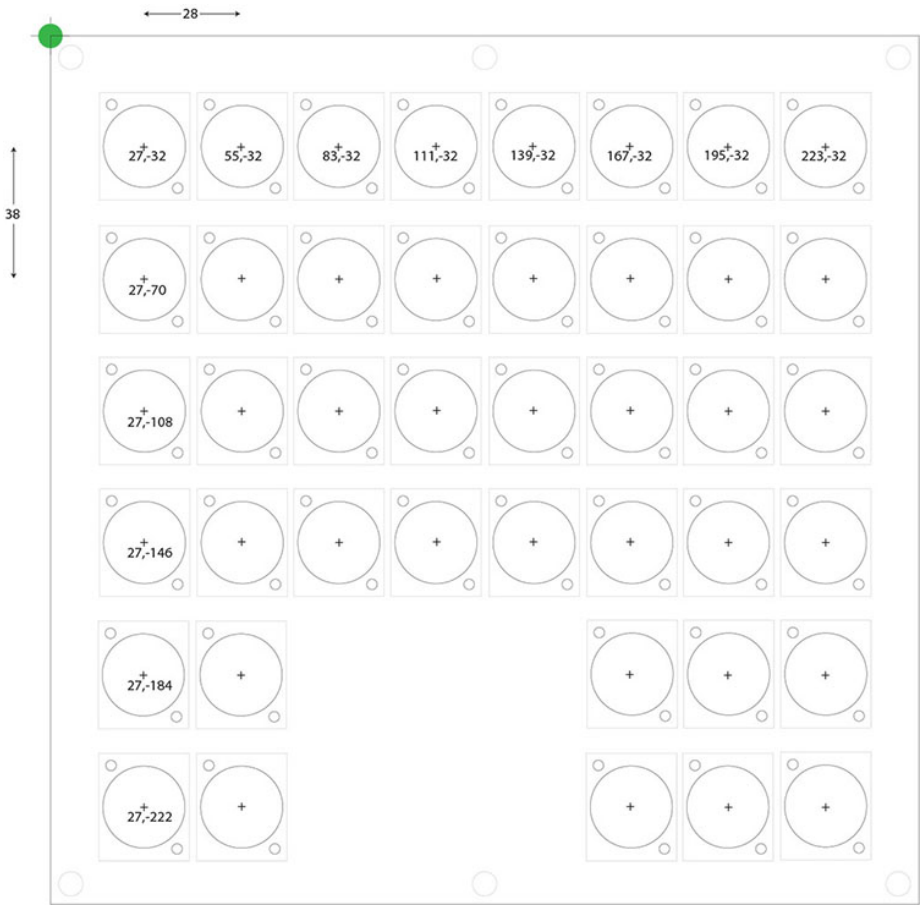


Fig. 7-13 Layout of holes for XLR sockets.

This job was machined by defining a subroutine to machine the holes for one socket. Another subroutine was created using instructions to call the first one eight times, to machine a complete row of eight sockets.

The individual subroutines must be defined individually, but any subroutine can be called from within any other subroutine (sometimes termed a ‘nested’ call).

Other subroutines were created for partially full rows, and they also used that first subroutine that machines the hole for a single socket. All these subroutines were called as required for each box lid.

Here are the subroutines used for the box lids.

Mach3:
O100 ***(subroutine to machine a single socket)***
← ***insert code to machine a single socket (given previously)***
M99
O110 ***(subroutine to machine a row of sockets)***
← ***same as commands in*** O110 ***above***
M99
O120 ***(subroutine to machine a partial row of sockets)***
← ***same as commands in O120 above***
M99

LinuxCNC:
(*Single socket*) O100 sub
← ***insert code to machine a single socket (given previously)***
O100 endsub
O110 sub ***(subroutine to machine a row of sockets)***
G0 X27
O100 call
G0 X55
O100 call
G0 X83
O100 call
G0 X111
O100 call
G0 X139
O100 call
G0 X167
O100 call
G0 X195
O100 call
G0 X223
O100 call
O110 endsub
(*Partial row of sockets*) O120 sub

```
G0 X27
O100 call
G0 X55
O100 call
G0 X167
O100 call
G0 X195
O100 call
G0 X223
O120 endsub
```

Note that subroutines 110 and 120 change the X coordinates as the Controlled Point moves from one socket position to another across a row, but do not change the Y coordinates of the current row. This is important because the individual rows differ by the Y coordinate and once that is set for a particular row, it should not change until the program moves on to the next row.

If the program had been written to machine holes down the columns, the X coordinate for any column would need to stay the same for each hole while the Y coordinate changed to locate the holes down the column.

In the main program, move to the start of a row position then call **O110** or **O120** to suit the row of sockets at that position. Repeat that action for each row in turn.

So a full set of three complete rows, one blank row and two partial rows would require a main program as follows.

LinuxCNC:

```
G0 X0 Y0 Z25
G0 X0 Y-32
O110 call
G0 X0 Y-70
O110 call
G0 X0 Y-108
```

```
O110 call
(No need to code for the blank row.)
(Just skip past it)
(Then do the two partial rows)
G0 X0 Y-184
O120 call
G0 X0 Y-222
O120 call
M30
```

Mach3:

```
G0 X0 Y-32
M98 P110
G0 X0 Y-70
M98 P110
G0 X0 Y-108
M98 P110
(No need to code for the blank row, just skip past it)
(Then do the two partial rows)
G0 X0 Y-184
M98 P120
G0 X0 Y-222
M98 P120
M30
```

In fact, this particular job was too large for the Y travel available on the mill, so the workpiece was mounted in a fixture consisting of a sheet of MDF and four locating pins that fitted the holes through the screw mounting lugs in the box lid (Figs 7-14 and 7-15). The main program was modified so that only the first four rows were machined. The workpiece was then turned around in the fixture and a second version of the program was used to machine the remaining rows, bearing in mind that the pattern of holes across each row was then

reversed. This all worked because the job was laid out symmetrically and because the socket holes were themselves symmetrical about their centres.

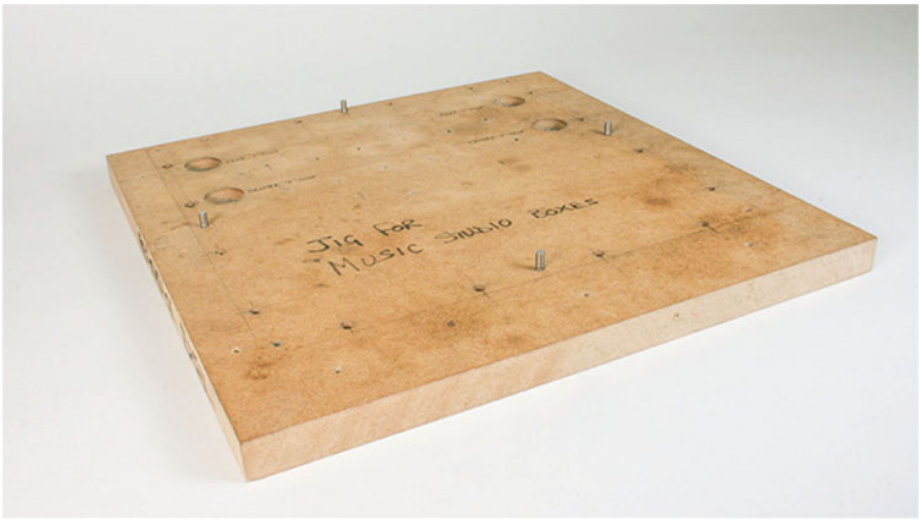


Fig. 7-14 Fixture to hold a large box lid.

Fig. 7-15 Fixture showing the lid mounted in position for machining.

Project 7.2

Tool Rack

Keeping frequently used tools in an easily reached position helps speed the work, as well as keeping the bench tidy. Fig. 7-16 shows a simple tool rack made from 30mm aluminium angle, 2mm thick, in which the tools are held in slots with semicircular ends. Slots have the advantages that the tool can be slid straight into the rack and there is no need for additional free space above the rack to allow tools to be lowered down into the rack.

Fig. 7-17 shows some dimensions of a tool rack that uses slots, but these can easily be modified so choose the number of slots and the dimensions of each slot to suit the tools.

While drilling the holes for the screws to fix the rack to the wall is not difficult, machining the slots needs more care.

Design Features

- The width of each slot gives an easy fit for the tool shanks but a secure base for the underside of the handles.
- The inner ends of each slot are semicircles and are positioned so that the handle of the tool is kept clear of the wall. The spacing of the slots allows a finger-width clearance on each side of the tool handle.
- Smaller tool racks can be made from thinner material, while larger racks will require thicker material.

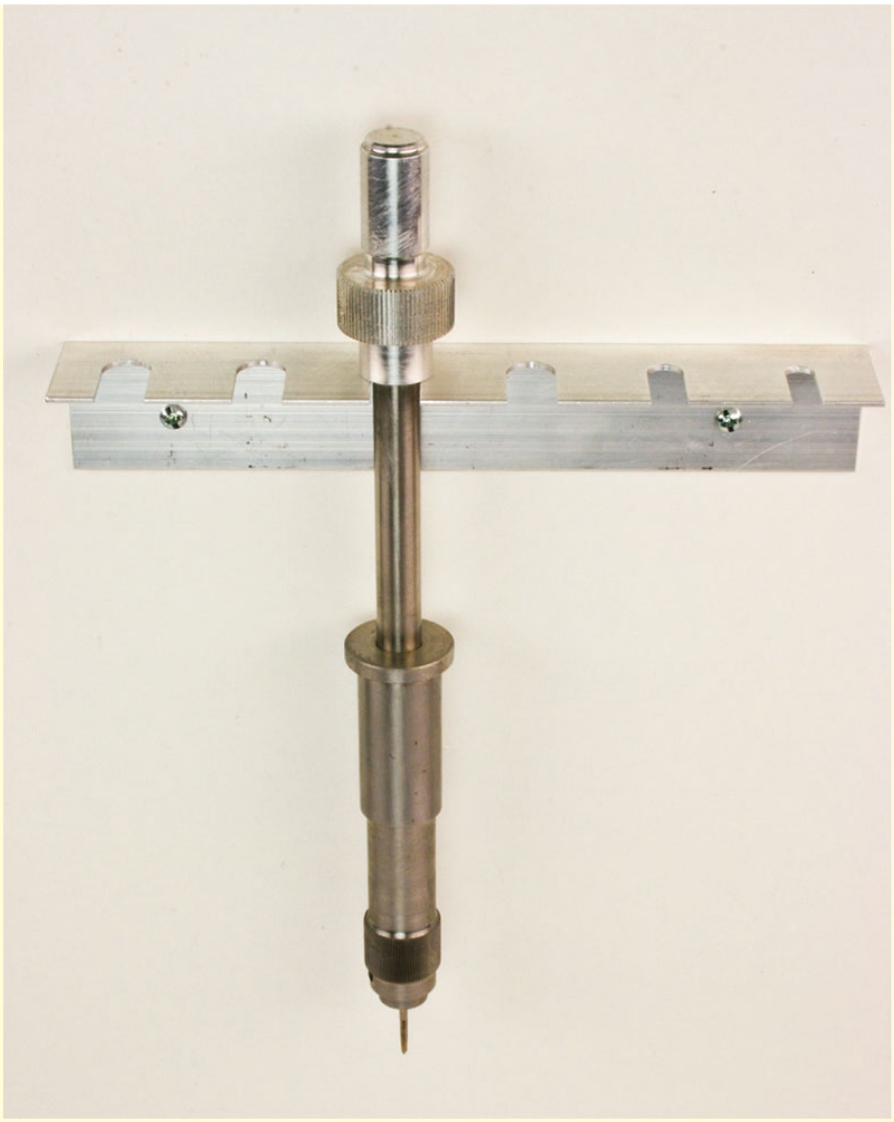


Fig. 7-16 A slotted tool rack.

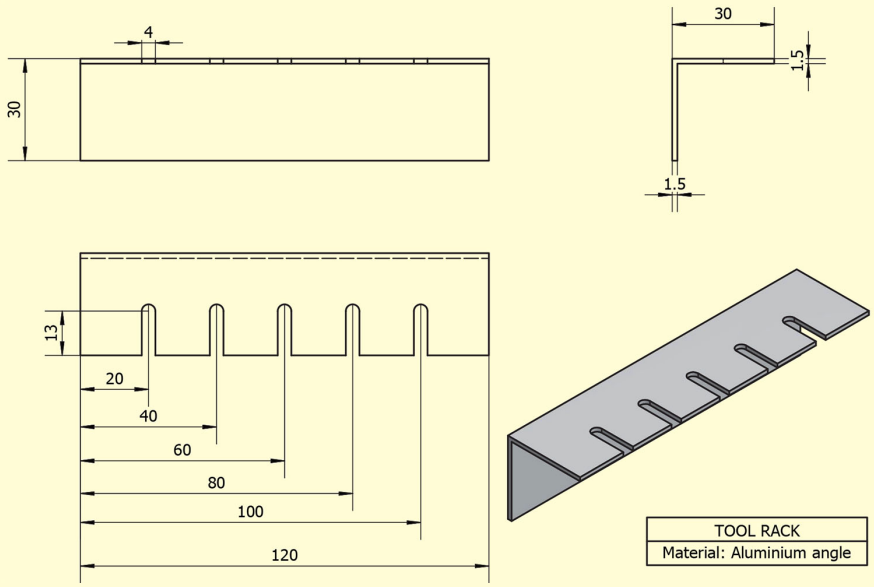


Fig. 7-17 Suggested dimensions for a slotted tool rack.

Material

- Use aluminium or a stiff plastic. The rack could be made of steel but would need to be painted.

Tool

Use an end mill or slot drill a little narrower than the width of the slot. This allows the tool to cut along each side of the slot in the same direction (on the way up one side, and returning down the other). With aluminium, you may find that removing the bulk of the material then ‘climb’ milling to final size produces the best finish. Using a cutter the same width as the slot will result in simultaneous cuts on both sides and this is unlikely to provide the finest finish.

Speeds and Feeds

Using a 6mm (¼in) end mill, the speeds and feed rates are shown in Table 7-2.

If you are using a specialist aluminium cutter, such as a polished router bit, speeds can be higher.

Slots in thin material are best machined with full-depth cuts, and the material needs to be supported firmly to prevent vibration or lifting of the relatively thin material.

Table 7-2

Material	Speed (rpm)	Feed rate (mm/min) [in/min]
Aluminium	2,000	50 [2]
Plastic	400	100 [4]
Steel	1,200	25 [1]

CLIMB MILLING

In conventional milling, the cutter travels on the right of the edge (when viewed from behind the cutter) and each tooth scoops into the material it is travelling towards, so the tooth and the material are effectively travelling in opposite directions.

When climb milling, the cutter rotates in the same direction but travels on the left of the edge, and each tooth cuts as the material travels past so tooth and material are travelling in the same direction.

Compared to conventional milling, the shape of the chip is different. In conventional milling, the chip thickness taken by each tooth is shallow at the start, thickening towards the end. In climb milling, the chip thickness is greatest at the start, reducing to a minimum towards the end.

Climb milling may show up deficiencies in the mill, particularly if there is backlash present in the leadscrews, but it can give an improved finish with shallow final cuts.

You may wish to use a lubricant such as white spirit, but wear a mask to avoid breathing the vapour.

Method

Define a subroutine:

- Create a subroutine to machine a slot, assuming the centreline of the slot is X0 and the open end of the slot is Y0.

Within your subroutine:

- Redefine the current point as X0 Y0.
- Include the instructions for machining the slot.
- Cancel the temporary coordinate offset, restoring the original coordinates.

If there are other slots of different widths, define a different subroutine for each width of slot.

In the main program:

- Set the X and Y origin to the front left corner of the rack and the Z origin to the top surface.
- Set the safe Z height above the height of any obstructing clamps.

For each slot in turn:

- Move to the slot position (X at the centreline and Y at the open end of the slot);
- Call the subroutine to machine that slot.

Same Shape, Different Size

The fan grille pattern in Fig. 7-18 consists of a series of interrupted squares that differ only by size. Rather than have to write separate

instructions for each square, we can use one subroutine to machine a typical square, but specify the length of the square when we use the subroutine by using a *parameter* (called an *argument* in LinuxCNC).

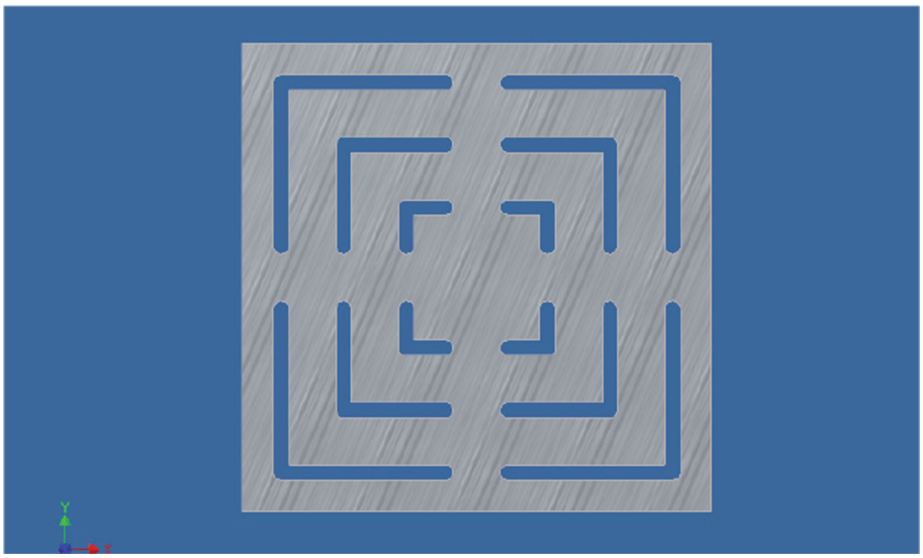


Fig. 7-18 A fan grill.

A parameter is a storage location. When a parameter is used, the program will go to that storage location and use the value it finds there. That means an instruction that refers to a parameter does not need to be rewritten if the value changes.

A parameter is created by using the # character followed by:

LinuxCNC:
an integer between 1 and 5399, but it is safest to use integers between 40 and 4999.

Mach3:
an integer between 1 and 10320, but it is safest to use integers between 40 and 4999.

So #41 is a parameter and so is #273. The = sign assigns a value to a parameter by putting that value in the appropriate storage location.

The command #41=50 puts the number 50 into storage location 41.

G0 Z#41 makes the Controlled Point move to Z50 because the contents of storage location 41 are 50.

Changing the value held in parameter 41 will change the value used, so that:

#41=26

G0 Z#41

makes the Controlled Point move to Z26.

Values held in parameters can also be used in arithmetic calculations, so that

#42 = [0.5*#41]

places half the value of parameter 41 into parameter 42;

#40 = [#52 + #63] places the sum of the values from parameters 52 and 63 into parameter 40; and

#97 = sin30 calculates the value of the trigonometric function sin30 and places it in storage location 97.

G0 X#43 Y#57 moves the Controlled Point to the position whose X coordinate value is found in storage location 43 and whose Y coordinate is found in storage location 57.

Note the use of square brackets to enclose the calculations. These are not always necessary, but are always wise. Enclose any calculation or calculation-within-a-calculation in square brackets, especially if the order of a calculation is not obvious to anyone who reads the program.

Although the software package will carry out the calculation in a specific order, it may be necessary to enclose parts of the calculation in brackets to ensure they are carried out in a known way.

Normally, parts of calculations are carried out in this order:

- anything in square brackets []
- multiplication * and division /
- addition + and subtraction –

For example: if location 73 holds the number 6:

CHOOSING PARAMETERS

Safe Parameters

Parameters #40 to #4999 should be safe to use in most systems, including Mach3 and LinuxCNC, but it is always wise to check the manual for your own software.

Local Parameters

Parameters 1 to 30 should be treated as having values that are *local* to the subroutine and may not retain their values in the main program. If you assign values to those parameters within the main program, they will not retain those values inside a subroutine. This can cause unexpected problems, so these parameters are best avoided unless you want to make use of this particular property.

Parameters numbered 31 or greater are global parameters that will retain their values everywhere in a program. In everyday use, it is safest to avoid parameters 1 to 30. In general, it is safest to use parameters from #40 onwards in your programs, unless you have good reasons for doing otherwise. #31 to #39 will also be global, but it is easier to remember to start at #40 as it is a nice round number.

System Parameters

Some parameters above #5000 are used to hold system information and should be avoided.

- 4 * #73 + 2 means calculate 4 times the contents of location 73, then add 2 to give 26 (that is, 4 times 6, then add 2).
- 4 * [#73 + 2] means add the contents of location 73 to the number 2, then multiply the answer by 4 to give 32 (that is, 6 + 2 = 8; then 4 times 8 = 32).

The interrupted square shown in Figs 7-19 and 7-20 is centred on (0, 0) and the slot can be cut using a 2mm diameter end mill. The code might be:

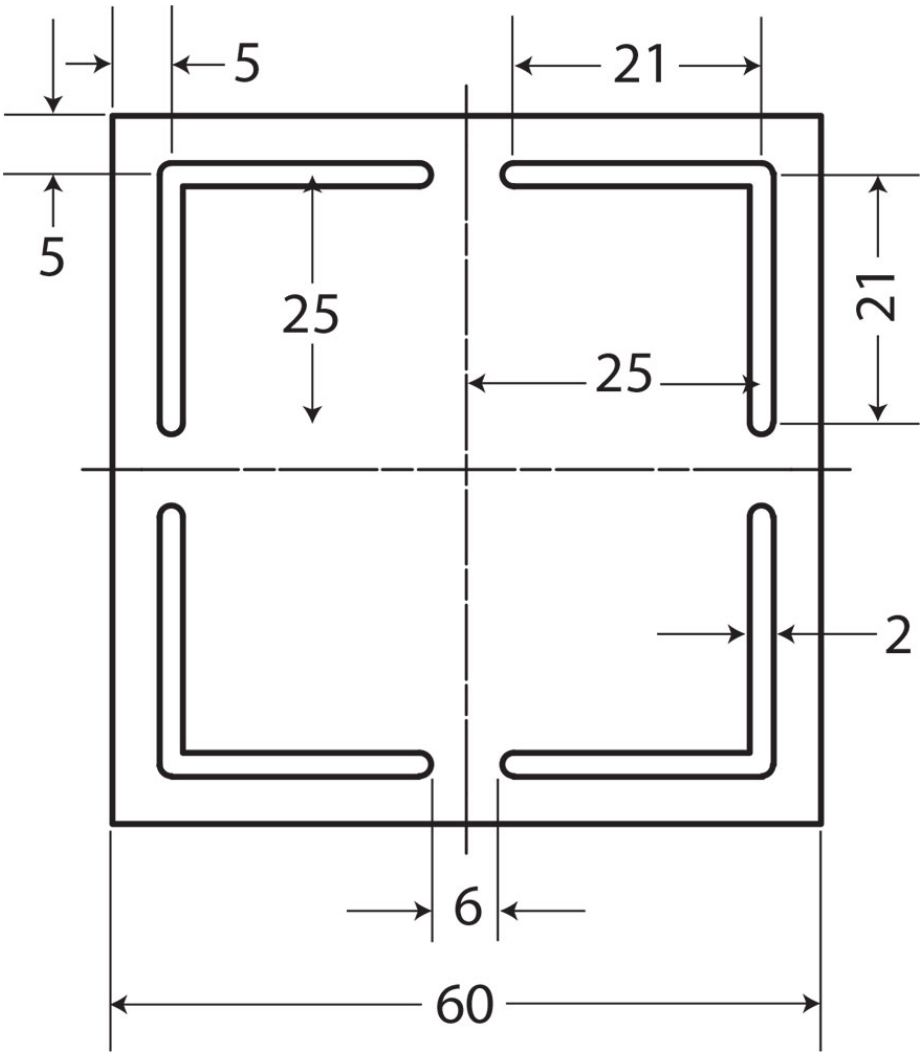


Fig. 7-19 Dimensions of an interrupted square.

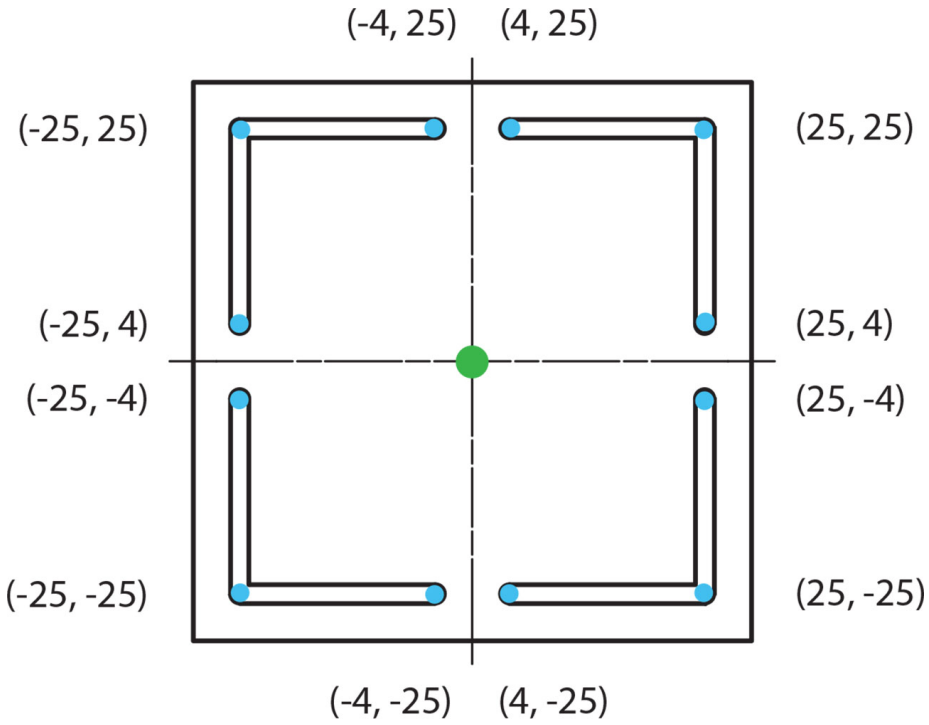


Fig. 7-20 Coordinates of cutter positions (shown in blue) for an interrupted square.

G0 Z1 (safe Z)
G0 X25 Y25 Z0.1 (Locating the Controlled Point above the centre line)
G1 X4 Y25 Z-1
G1 X25 Y25
G1 X25 Y4
G0 Z1
(Repeat this sequence for the next slot)
G1 X-25 Y25 Z0.1
G1 X-4 Y25 Z-1
G1 X-25 Y25
G1 X-25 Y4
G0 Z1
(and the next)
G0 X-25 Y-25 Z0.1
G1 X-4 Y-25 Z-1
G1 X-25 Y-25

G1 X-25 Y-4
G0 Z1
(and the last)
G0 X25 Y-25 Z0.1
G1 X4 Y-25 Z-1
G1 X25 Y-25
G1 X25 Y-4
G0 Z1
G0 X0 Y0

For an interrupted square of a different length, a new set of instructions would be required.

Using a parameter, a subroutine can be created that refers to the length of the square instead of a fixed value. When the subroutine is called, it will use the value stored in the parameter. Here is a subroutine that machines an interrupted square, as shown in Fig. 7-21 (where the blue circles indicate cutter positions). It obtains the length of the square from parameter 101. It also uses parameter 102 to store half the length of the square, and parameter 103 to store the negative of the contents of parameter 102. If the length of the square is 50, measured between the centrelines of the slots, #102 will contain 25 and #103 will contain −25 so that, for example, the four corners will have coordinates: bottom right (25, −25) or (#102, #103); top right (25, 25) or (#102, #102); top left (−25, 25) or (#103, #102); and bottom left (−25, −25) or (#103, #103). Changing the value stored in #101 will result in new values being calculated for #102 and #103, and new coordinates being used for the points defined by those parameters.

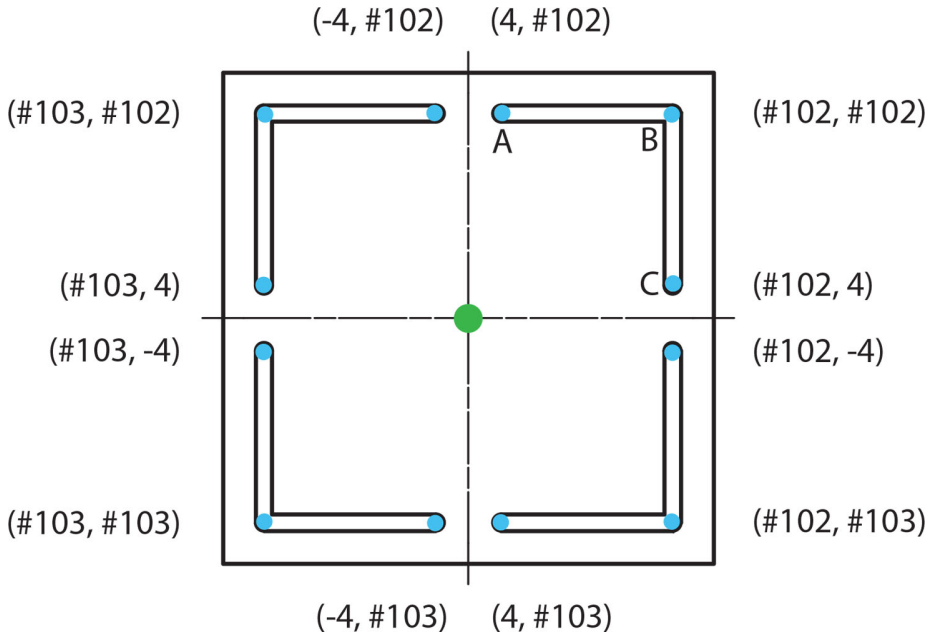


Fig. 7-21 Parameters used to define an interrupted square.

O500 sub (Mach3: **O500**)

Mach3: remember that the subroutine definition goes after the end of the main program.

#102 = [#101/2] (defines the value stored in parameter 102 as half the value stored in parameter 101)

#103=[-1*#102] (defines the value stored in parameter 103 as the negative of the value stored in parameter 102

G0 Z1 (safe Z)

G0 X#102 Y#102 Z0.1 (where #102 is half the length of the square)

G1 X4 Y#102 Z-1 (ramp into the work along one arm of the slot)

G1 X#102 Y#102 (cut back to the corner at that level)

G1 X#102 Y4 (machine the other arm of that slot)

G0 Z1

(Repeat this sequence for the next slot)

G0 X#103 Y#102 Z0.1
G1 X-4 Y#102 Z-1
G1 X#103 Y#102
G1 X#103 Y4
G0 Z1
(and the next)
G0 X#103 Y#103 Z0.1
G1 X-4 Y#103 Z-1
G1 X#103 Y#103
G1 X#103 Y-4
G0 Z1
(and the last)
G0 X#102 Y#103 Z0.1
G1 X4 Y#103 Z-1
G1 X#102 Y#103
G1 X#102 Y-4
G0 Z1
G0 X0 Y0
O500 endsub (Mach3: **M99**)

In the main program, define the value stored in parameter 101 using:

#101=50

then call the subroutine

O500 call (Mach3: **M98 P500**

For a pattern of three interrupted squares with lengths 50, 40 and 30, the main program would look like this:

LinuxCNC:
#101=50
O500 call
#101=40
O500 call
#101=30
O500 call
M30

Mach3:

```
#101=50
M98 P500
#101=40
M98 P500
#101=30
M98 P500
M30
```

Parameters make it easy to write general-purpose subroutines for machining commonly used shapes. There is a good deal more to learn about the advanced use of parameters, but these basic principles will take you a long way.

LOOPS

Loops can be used to repeatedly execute a series of instructions, including calls to subroutines. Loops are programmed differently in Mach3 and LinuxCNC. Combining loops with subroutines, parameters and incremental distances allows the depth of cut to be varied each time around the loop, making this a powerful programming tool.

A Basic Loop Structure

Mach3: The number of times a subroutine is to be carried out can be stated as part of the Call.

The full syntax of the M98 ***command in Mach3 is***

M98 O<subroutine number> L<number of times to go round the loop>

So M98 O120 L8 ***will carry out the subroutine O120 eight times.***

LinuxCNC: The repeat ***command is used to define the number of times the commands inside a loop are to be carried out. The structure of a loop is rather like the structure of a subroutine, except loops do not have to be defined before they are used.***
O160 repeat [8]
...
O160 endrepeat
will repeat any instructions placed inside the loop, eight times.
The O command is used just as in a subroutine, along with the repeat and endrepeat commands, to define the beginning and end of the loop.
The value in square brackets following the repeat command specifies the number of times the loop is to be repeated.

Using Incremental Mode to Control Depth of Cut

So far, we have used absolute coordinates, with all distances being measured from the current coordinate system as set by the **G54** offsets created by touching off. It is possible to choose to use incremental distances instead, measured from the current position of the Controlled Point.

In absolute coordinates, **G0 X1 Y2 Z3** will move the Controlled Point to the coordinates (1, 2, 3) measured from the work origin.

In incremental distance mode, **G0 X1 Y2 Z3** will move 1 unit in X, 2 units in Y and 3 units in Z relative to the current position of the Controlled Point. This is like temporarily setting a new origin at the current position of the Controlled Point, and moving from there.

To begin using incremental distance mode, use the **G91** command. To revert to absolute distances, use **G90**. Incremental distances can be used as a simple way of increasing depth.

Mach3:
G0 X0 Y0
G0 Z0
G91 ***(Begin using incremental distances)***
M98 O250 L10
G90 ***(Revert to absolute distance mode)***
M30
O250
G1 Z−0.2 ***(Add −0.2 to the current value of Z)***
G1 X0.5 ***(Add 0.5 to the current value of X)***
M99

LinuxCNC:
O250 sub
G1 Z-0.2 ***(Add −0.2 to the current value of Z)***
G1 X0.5 ***(Add 0.5 to the current value of X)***
O250 endsub
G0 X0 Y0
G0 Z0
G91 ***(Begin using incremental distances)***
O300 repeat [10]
O250 call
O300 endrepeat
G90 ***(Revert to absolute distance mode)***
M30

Adding a Parameter to Control Depth of Cut

Fig. 7-22 shows one method of altering the depth of cut each time around a loop while staying in absolute distance mode. This subroutine uses #40 to hold the current depth of cut. Each time around the loop, the value stored in that parameter is increased by the depth of cut.

So, to take ten cuts of 0.25mm each to cut to a total depth of 2.5mm, begin with a basic subroutine.

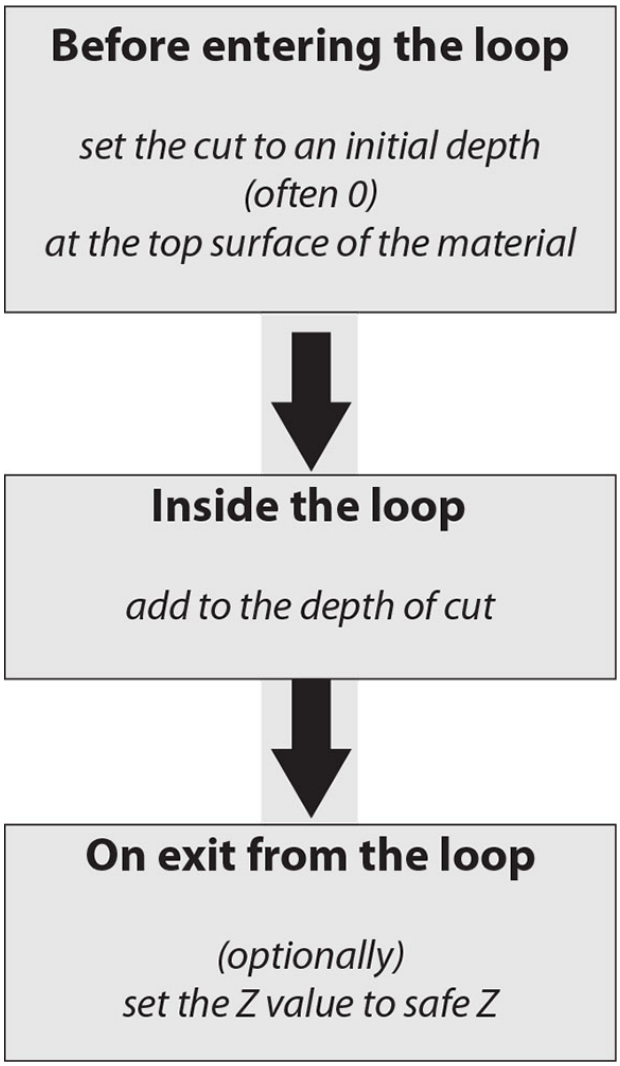


Fig. 7-22 Stages in constructing a loop to apply a cut.

O350 sub (Subroutine to cut a slot) (Mach3: **O350**)

#40 = [#40−0.25] (Make the depth of cut 0.25 lower)
G1 Z#40
G1 X100 Y0
G0 Z25 (Safe Z)
G0 X0 Y0
O350 endsub (Mach3: **M99**)

Then set an initial depth of cut (often 0, but this will depend on the job), and execute the loop.

Mach3:
#40 = 0 (Set initial depth of cut to 0)
M98 O350 L10 (execute the loop ten times)
#40 = 0 (Set depth of cut parameter to 0 again, for safety)

LinuxCNC:
#40 = 0 ***(Set initial depth of cut to 0)***
O160 repeat [10]
O350 call
O160 endrepeat
#40 = 0 ***(Set depth of cut parameter to 0 again, for safety)***

Using Loops, Parameters and Arithmetic to Machine a Rectangle

Fig. 7-23 shows a rectangle defined in terms of parameters #80 to #85, and the following code calculates some additional parameters, #90 to #95, to act as coordinates, then uses a subroutine to machine that rectangle by making use of those parameters.

The program first sets up the values stored in the additional parameters and then calls subroutine 300, which does the actual machining. Fig. 7-24 shows the co-ordinates of important points on the toolpath in terms of those additional parameters.

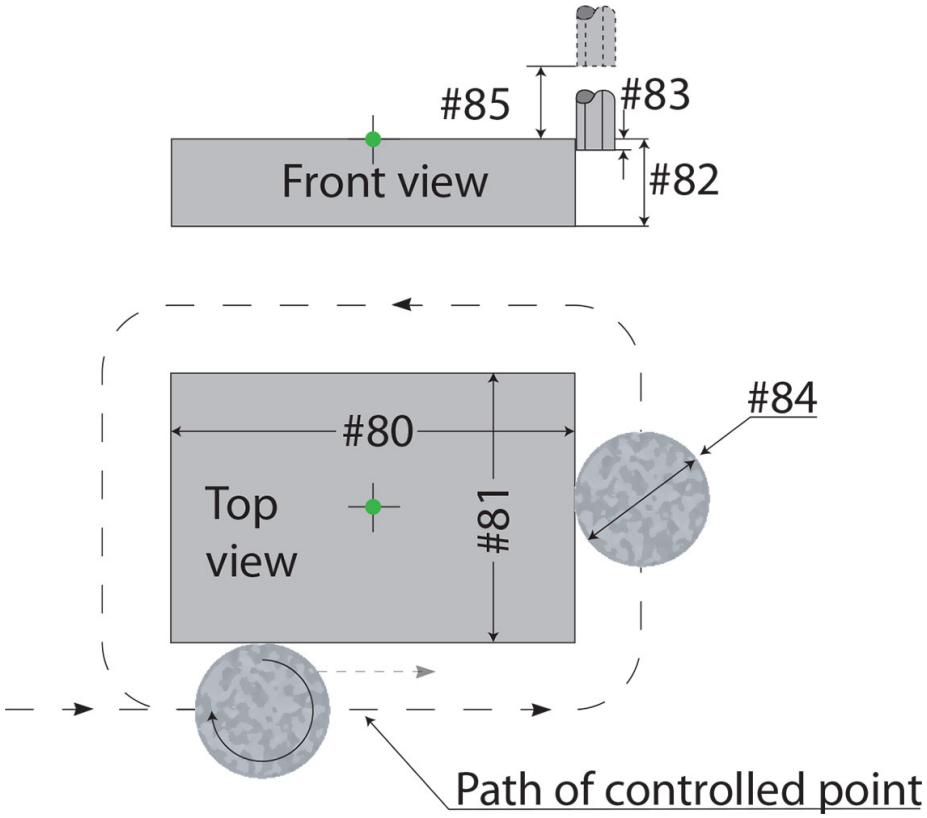


Fig. 7-23 A rectangle defined in terms of parameters.

Note that the function **FIX[ ]** rounds a value down to the next lowest integer and is used to calculate the number of passes of the cutter. In the subroutine, the Controlled Point follows the toolpath around the outside of the rectangle, and that toolpath takes account of the diameter of the cutter.

(Cutter runs outside the rectangle)
(The origin is at the centre of the rectangle)
(Z=0 at the top surface)
(Position the Controlled Point above the centre of the rectangle)
(before using this program)
(#80 holds the length of the rectangle)
(#81 holds the width of the rectangle)
(#82 holds the final depth of cut)
(#83 holds the depth of cut at each pass)
(#84 holds the cutter diameter)
(#85 holds the Safe Z height)

(#90 will hold half the rectangle's length)
(#91 will hold half the rectangle's width)
(#92 will hold the current depth of cut)
(#93 will hold the number of passes)
(#94 will hold half the toolpath length)
(#95 will hold half the toolpath width)
(Subroutine to cut around the periphery)
O300 sub (Mach3: **0300**)
(Position the cutter at F)
G0 X[-1*#90] Y[-1*#95]
(Cut vertically to set the depth)
G1 Z#92
(Cut the straight part of the length, to G)
G1 X#90 Y[-1*#95]
(Quarter circle around the corner, to H)
G3 X#94 Y[-1*#91] I0 J#84
(Cut the width, to K)
G1 Y#91
(Quarter circle from K to L)
G3 X#90 Y#95 I[-1*#84] J0
(Along to M then round to N)
G1 X[-1*#90]
G3 X[-1*#94] Y#91 I0 J[-1*#84]
(Down to P, then around to F)
G1 Y[-1*#91]
G3 X[-1*#90] Y[-1*#95] I[#84]
J0
(Increase the depth for the next cut)
#92 = [#92+#83]
O300 endsub (Mach3: **M99**)
(Main program)
#80 = 100 (Set the length)
#81 = 50 (Set the width)
#82 = −22 (Set the final depth of cut)
#83 = −5 (Set the depth at each pass)
#84 = 5 (Set the cutter radius)

(Set the Safe Z height)
#85 = 10
(Half the length of the rectangle)
#90 = [#80/2]
(add on the radius of the cutter)
(to find half the length of the toolpath)
#94 = [#90+#84]
(Half the width of the rectangle)
#91 = [#81/2]
(add on the radius of the cutter)
(to find half the width of the toolpath)
#95 = [#91+#84]
(Set the initial value of)
(the current depth of cut)
#92 = #83
(Calculate the number of passes)
(to get as close as possible)
(to the final depth)
(without cutting below the final depth)
#93 = FIX[#82/#83]
(Make this point 0, 0 temporarily)
G92 X0 Y0
(Set the feed rate)
F100
(Now loop around the periphery)

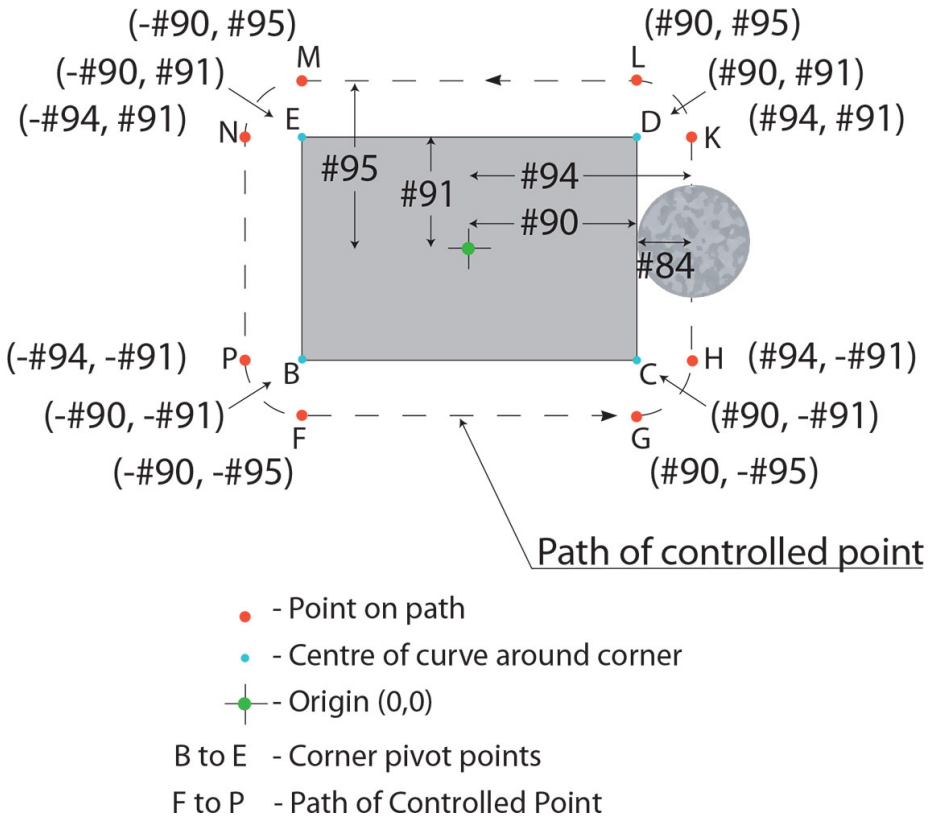


Fig. 7-24 Toolpath around a rectangle, showing important points expressed in terms of parameters.

LinuxCNC:
O400 Repeat [#93]
O300 call *(calls the subroutine to cut around the periphery)*
O400 endrepeat

Mach3:
M98 P300 L#93 (calls the subroutine to cut around the periphery, #93 times)

Project 7.3

Socket Set Accessory Rack

Socket sets can be organized easily using little spring clips mounted on long strips that can be secured to a wall, but there is no provision for accessories like screwdriverstyle handles, extension bars or swivelling handles. These need a rack of their own, and most will fit into a set of suitably sized holes (Fig. 7-25).

Fig. 7-25 A socket set accessory rack.

Fig. 7-26 shows dimensions of a rack that uses holes large enough for the main parts of the accessories to pass through, but small enough so that the end part of the accessory is held. Hole diameters and positions can easily be modified to suit your own tools.

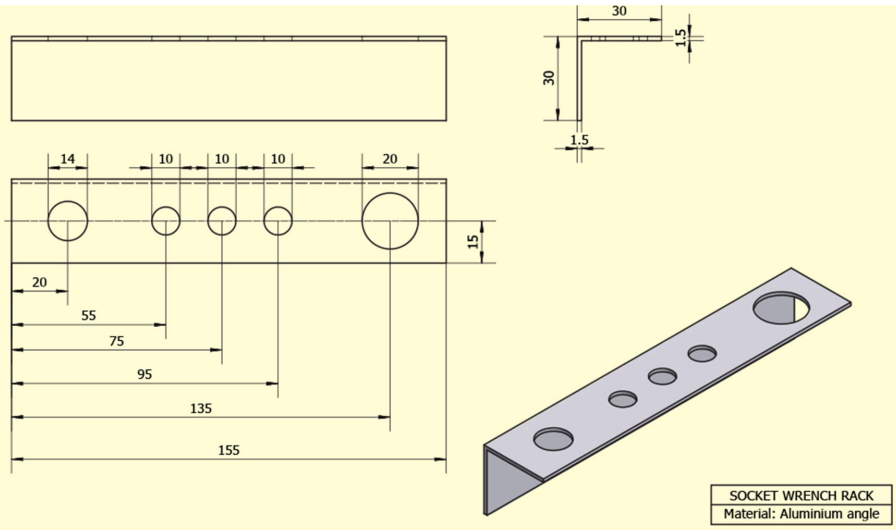


Fig. 7-26 Suggested dimensions for the accessory rack.

Design Features

- The diameter of each hole gives an easy fit for the tool shanks but a secure base for the underside of the handles.
- The spacing between holes needs to be large enough for fingers to grip the accessories to remove them from the rack.

Material

- Use aluminium or a stiff plastic. The rack could be made of steel, but would need to be painted.

Tool

Use an end mill or slot drill smaller in diameter than the smallest hole. Cutters designed for the material being machined will cut more efficiently, but may be expensive or difficult to obtain. In that case, a standard end mill or slot drill for steel will do.

Speeds and Feeds

Using a 6mm (¼in) end mill, use the speeds and feeds in Table 7-3.

Table 7-3

Material	Speed (rpm)	Feed rate (mm/min) [in/min]
Aluminium	2,000	50 [2]
Plastic	400	
Steel	1,200	25 [1]

If you are using a specialist aluminium cutter such as a polished router bit, speeds can be higher. When machining aluminium, you may wish to use a lubricant such as white spirit, but wear a mask to avoid breathing the vapour.

Method

(Optional) Define a subroutine for each diameter of circle:

- Create a subroutine to machine a circular hole, assuming the centre of the circle is at X0 Y0.

Within your subroutine:

- Redefine the current point as X0 Y0.
- Include the instructions for machining the circle.
- Cancel the coordinate offset, restoring the original coordinates.

In the main program:

- Set the X and Y origin to the front left corner of the rack, and the Z origin to the top surface.
- Set the safe Z height above the height of any obstructing clamps.

For each hole in turn:

- Move to the hole position.
- Call the subroutine to machine that hole.

Extending your skill:

- Create a single subroutine to machine a hole of any diameter by using a variable to represent the size of the hole.
- Extend this to create a single subroutine to machine a hole of any diameter using any size of cutter by using one variable to represent the size of hole and a second variable to represent the diameter of the cutter.

(Final pass at full depth)
(Make the current depth of cut = the final depth) **#92 = #82**
(Then perform the final pass just once)
O300 call (Mach3: M98 P300)
(Go back to the temporary origin)
G0 X0 Y0
(Now restore the original coordinates)
G92.1
M30

Making Decisions

In LinuxCNC there are other types of loops, and there are conditional logic commands that allow branching if a condition is true or false. These additional commands extend the capability of LinuxCNC considerably, allowing complex movements to be controlled efficiently using a relatively small number of commands. There are no equivalent commands in Mach3.

Both the While/Endwhile and the Do/While loops repeat while a condition is true. They differ only in the position of the test within the loop.

In the While/Endwhile loop, the test is carried out at the beginning of the loop, before any of the instructions inside the loop have been executed. If the result of the test is false, the program exits the loop

without carrying out the other instructions. This means the instructions inside the While/Endwhile loop may never be executed.

In the Do/While loop, the test is carried out at the end of the loop, after the instructions in the loop have been executed, so this means the instructions inside the Do/While loop will always be executed at least once.

These two loop structures can be used rather like a repeat/endrepeat loop, but without needing to state the number of repeats of the loop.

For example: if #40 is the Z cut, and the subroutine O350 is as shown on page 92, incrementing Z by −0.25 each time it is called, the following loop will repeat until Z is −5 or less. It will not take a cut when Z = −5, so the last cut will be when Z = −4.75.

MATHEMATICAL COMPARISONS

Mathematical expression	Meaning
EQ	Equal to
NE	Not equal to
GT	Greater than
GE	Greater than or equal to
LT	Less than
LE	Less than or equal to

#40 = 0
O160 while [#40 gt −5] (loop until depth of cut = −5)
O350 call
O160 endwhile

This next loop will repeat until Z is −5 or less. It will take a cut when Z = −5.

#40 = 0
O160 do
O350 call
O160 while [#40 gt −5]

Note that in the last two examples, the repeat loops could have contained all the instructions from the subroutine O350 inside the loops, so that a separate subroutine need not be used. Using a subroutine means that same subroutine can be called from any other part of the program if necessary.

Conditional tests are based on the IF statement and they can be used to choose between one course of action and another. The structure is: if, else, endif. If a condition (or a test) is true, one set of commands is executed. If the same condition (or test) is false, a different set of commands is executed.

If #40 represents a depth of cut, for example:

```
O360 if [#40 GT -3]
G1 Z#40          ← do these commands if
G1 X100 Y0       ← #40 is greater than -3
O360 else
G0 Z20 (safe Z)  ← do this if #40 is not
                   greater than -3
O360 endif
```

Using Predefined Subroutines

Subroutines are so useful that a substantial part of a program might consist of several predefined subroutines to mill common shapes, and the program might simply be completed by adding some move commands and setting parameters before calling the subroutines required. Subroutines to create rectangles and rectangular pockets, circles and circular pockets, cut-outs for standard plugs and sockets, drilling patterns for holes in rectangular or circular patterns, and many others, would be a useful inclusion in any program. If particular subroutines are not required in a particular program, they can remain in the program but will not be called.

These subroutines could all be present in a ‘skeleton’ program that could be loaded whenever a program is to be written and the rest of the code could be added as required. This might speed the process of program writing. It might also aid debugging, because the preloaded subroutines will have been tested before inclusion in a program (or should have been).

Using a Library Subroutine from a File

An alternative approach to program writing would be to store a library of subroutines as individual files in a directory, then include a command inside the program to load and use specific subroutines from that library as they are required. This can be done using a variation of the command normally used to call a subroutine held within a program.

Where the program cannot find a subroutine within the program, it will automatically look for a named subroutine held as a file in a directory on the computer’s hard drive.

LinuxCNC: Two variables need to be set in the INI file: PROGRAM_PREFIX and SUBROUTINE_PATH need to be set to the path of the directory where the subroutine files are stored. This may have been done as part of the set-up of your INI file. If not, the file can be edited in a text editor and re-saved.

Create the subroutine and save it in the folder specified in your INI file.

Creating the subroutine in a text editor is fine, but the subroutine file needs to be saved with a .ngc extension (which you can enter manually after the file name). For example:

O<rectangle> sub
(Rectangle subroutine)
G0 X0 Y0 Z0

```
G1 Z-1
G1 X20
G1 Y20
G1 X0
G1 Y0
G0 Z10
O<rectangle>   endsub
M2
```

Save that file using the name rectangle.ngc Note that the file name and the name of the subroutine use lowercase letters. Within the main program, call the subroutine using

```
O call
```

Mach3: The M98 ***command can refer to a file name instead of a subroutine reference number. In that case, the software will go to the named file and use the subroutine it finds there and then return to the main program.***

Create a subroutine and save it in the Subroutines folder (in the Mach3 directory), for example

rectangle.tap

Note that the subroutine does not need an O command at the start, but it does need an M99 ***at the end. Creating the subroutine in a text editor is fine, but it must be saved with a .tap extension (which you can enter manually after the file name). For example:***

```
(Rectangle subroutine)
G0 X0 Y0 Z0
G1 Z-1
G1 X20
G1 Y20
G1 X0
G1 Y0
G0 Z10
M99
```

Save the file using the name
rectangle.tap
Note that the file name and the name of the subroutine use lowercase letters.
Within the main program, call the subroutine using
M98 P(rectangle.tap)
As with other subroutines, the M98 ***command may also include the L parameter to control the number of times the subroutine is called, for example***
M98 P(rectangle.tap) L2

Using Wizards

Prewritten subroutines are often known as **wizards**, a term that suggests these work by magic. Wizards have been around almost since the beginning of computer programming, and are sometimes supplied as part of a CAD/CAM package. Collections of wizards are available for several of the popular CNC programming packages and languages; some free and some requiring additional payment. These sometimes take the form of subroutines you can incorporate in your own program, but some collections of wizards are separate programs that act as code generators that create blocks of code for specific operations and then put that code into a CNC program. So a ‘rectangle’ wizard may ask you for the length and breadth of the rectangle and its starting position, and may then put the code required to machine that rectangle into a CNC program (or, if you are running the wizard within a CNC program, it will put the G code into the program execution window). Using wizards effectively and safely does require that you understand exactly what the wizard will do and exactly how the code will attempt to machine a shape.

There is a point of view that a CAM program does the work that wizards might do by translating a drawing into G code, but some wizards still have their uses, especially where a full CAM program is not available.

DOING THESE JOBS USING A CAM PROGRAM

A CAM program like Cut2D takes a different approach to this kind of job.

Firstly, it does not care whether a particular pattern is repeated or not. It does not use subroutines. Instead, it simply calculates the movements of the Controlled Point that are required for each feature. CAM software does not attempt to minimize the size of the program, so it may use hundreds of lines of code instead of a couple of dozen.

To machine the pattern of holes shown in Fig. 7-1, simply draw the holes (as circles) in a CAM program, then use the Pocket tool to create a toolpath. If the diameter of the end mill being used is smaller than the diameter of the smallest hole, the CAM program will generate a suitable toolpath.

Note, however, that the program will not take account of the presence of a rotary table. It simple works out the toolpath required to cut the holes from a sheet of material clamped to the bed of the mill. If the material is held on a rotary table, it will treat that as a stationary fixture and will machine all the holes without rotating the table. This may limit the overall size of any pattern to the available work envelope created by the maximum travel of the slides.

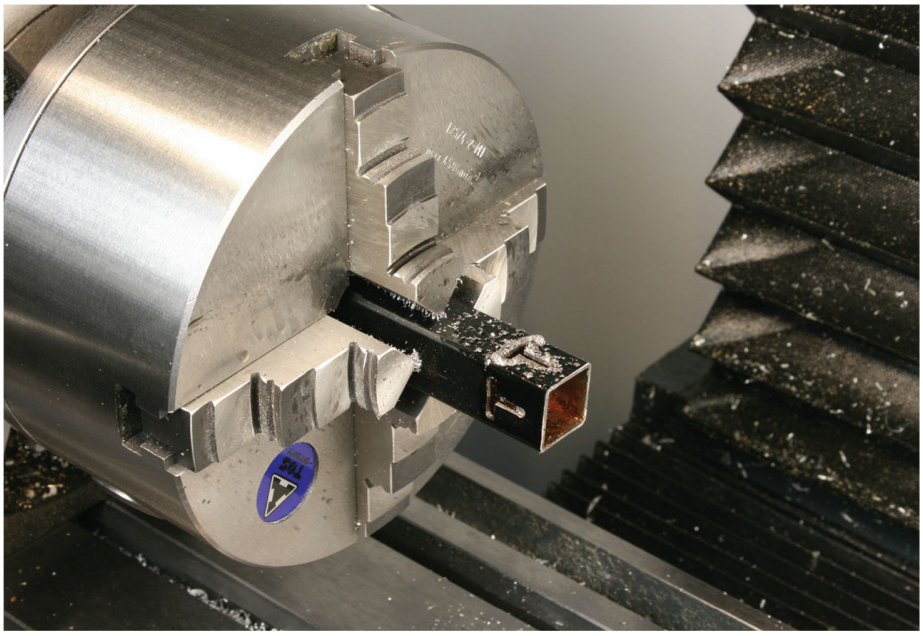


Fig. 7-27 A tube requiring a T-shaped slot on each face.

The set of sockets shown in Fig. 7-6 is treated in the same way. The CAM program does not attempt to identify repeated patterns of holes; it simply works out the toolpath for each individual hole at the position at which it is drawn. It does not use G92 commands.

It is sometimes useful to be able to use a CAM program to generate code for a complex shape and then to use that code within your own program. One way to do that is to put the CAM-generated code in a subroutine. Calling the subroutine causes the shape to be machined.

The tube shown in Fig. 7-27 requires a set of little T-shaped slots on all four faces. One way of doing that would be to use a CAM program to generate the code for a program to machine a single slot on one face of the tube; then remount the tube with another face upwards and run the program again. Repeat the process for all four faces. This involves four set-ups.

Mounting the same tube on a rotary table avoids having to remount the tube at any time. The method is to machine the first slot, rotate the table and machine the second slot and so on.

To do that, you could run the program, use MDI mode to turn the table, then run the program again and so on. Or you could take the code generated by the CAM program and put it into a subroutine within your own program as shown.

LinuxCNC:
<<Initialization block>>
O100 sub
← put the code from the CAM program here
O100 endsub
G0 A0
O100 call
G0 A90
O100 call

```
G0 A180
O100 call
G0 A270
O100 call
M30
```

```
Mach 3:
<<Initialization block>>
A0
M98 P100
A90
M98 P100
A180
M98 P100
A270
M98 P100
M30
O100 sub
← put the code from the CAM program here
M99
```

You may wish to edit out the initialization section right at the start of the CAM-generated code and create your own initialization section at the start of the program. Or you may wish to copy the initialization section from the CAM program into your own program.

Multiple parts can be made using fixtures such as this one, which is used to make six clock wheels at a time.

8 Making Multiple Parts

In this chapter you will learn about:

- how to create a program to make multiple copies of a single part;
- how to nest many different parts within a single workpiece.

USING A FIXTURE

A fixture is a device for holding a workpiece and usually provides a handy, secure and repeatable way of holding workpieces that will all be located in the same position. Fig. 8-1 shows a fixture used when machining a spacing piece for a security bracket. It holds one workpiece at a time, so the workpiece is loaded into the fixture, machining takes place, the workpiece is unloaded from the fixture and the whole cycle is repeated for the next workpiece. The lower strip in Fig. 8-1 is a sacrificial piece to keep the cutter from touching the surface of the fixture.

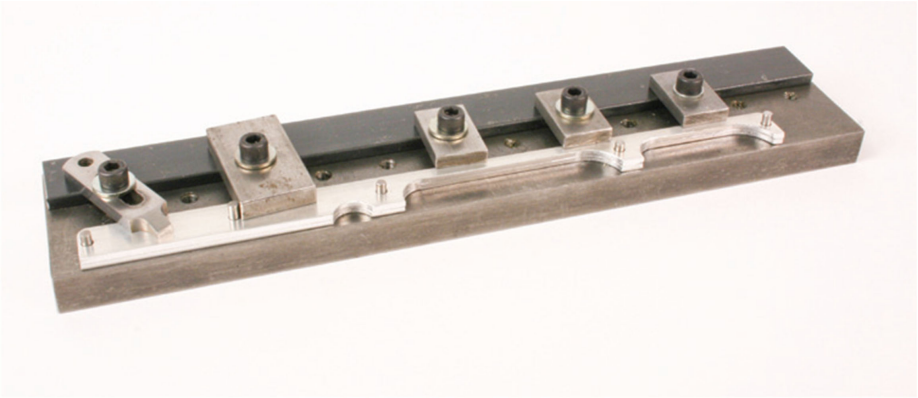


Fig. 8-1 Fixture for holding material to make part of a security bracket.

On a CNC machine, a fixture ensures that the origin is always in a known position in relation to the workpiece so that it is not necessary to reset the origin for each workpiece. Simply mount the fixture in the machine, put the first workpiece in the fixture and then set the origin. Because the fixture holds all the workpieces in the same position, the origin will be the same for each workpiece. When a fixture is used to hold workpieces one at a time, the same CNC program is run for each workpiece.

Fixtures can be used to hold material so that several finished items can be machined from one piece of material and a single block of material can also be used as if it were a fixture. In Fig. 8-2, thirty-two identical items are shown machined from a single block, which acts rather like a fixture. Sixteen items are machined into the top, then the block is flipped over and a further sixteen are machined in the bottom.

Fig. 8-2 Clock batons cut as a batch from the top and bottom faces of a single block of material, which acts as its own fixture.

LOCATING WORK IN A FIXTURE

The purpose of a fixture is to hold work securely and repeatably, so that when another workpiece of the same shape and size is placed in the fixture, it will occupy exactly the same position.

Straight edges and dowels are useful for locating against sides or edges and other features already on a workpiece. Straight edges can locate against flat faces but dowels are more flexible and can locate against straight edges, curves, or individual features of a workpiece.

On a workpiece that already has some machined features, it is important to consider which of those features should be used as a reference for the remaining machining. Existing bores, adjacent machined faces or machined projections might be important features. Bores can be located on a matching pin; machined faces can be placed against vertical pins, and machined projections may need to be located against the end face of a horizontal pin held above the base of the fixture. A general purpose fixture may be a grid of reamed holes which can carry pins located to suit the edges of the work. Work must be held securely in the fixture, and any clamps should be easy to operate and positioned so that they are out of the path of the cutter. Clamps are often incorporated into the design of the fixture.

Project 8.1

Small Fixture Plate for the Vice

Holding small fixture plates (Fig. 8-3) in the vice solves two problems.

Firstly, fixture plates for small parts need not be large, but their small size is limited by the fact that if a plate is to be securely located on the mill table, it must be large enough to be gripped by some decent clamps.

Secondly, benchtop mills often suffer from a range of movement that does not allow a short cutter to reach the table, as the mill runs out of Z travel near the bottom of its range. The mill is usually well able to reach work held in a vice, though, so if a small fixture plate can be held in the vice, it places it at an ideal height.

Although it is easy enough to grip a plate in the vice, with its top surface lying horizontally, the plate needs to be accurately set horizontal each time it is used, and that can be time-consuming.

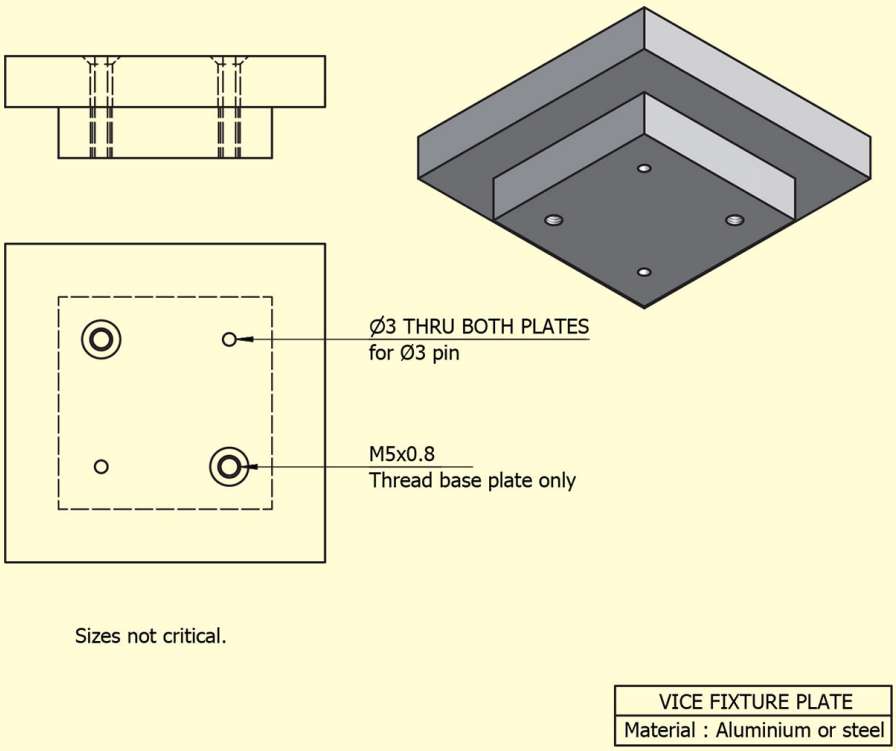


Fig. 8-3 A fixture plate designed to be held in a vice.

Adding a smaller plate or a small block under the main fixture plate eliminates this problem and makes set-up quick and easy. The fixture plates are usually quite small, so it is useful to have several available. They are quick and easy to make. Fig. 8-3 shows a typical plate, with the smaller plate underneath.

The vice jaws are opened and the underside of the upper plate is dropped on to the tops of the vice jaws. The jaws are then closed on the smaller plate underneath.

When the fixture is being made, it is gripped in position and a skim is taken off the top of the top plate, so that surface is guaranteed to be horizontal every time the plate is gripped in the vice. None of the dimensions of a fixture plate of this kind are critical. Use what you have to hand, as long as it allows the work to be held securely.

The thicker the top plate, the better, so that there is enough depth to tap holes and to prevent excessive flexing. The thicker the upper plate, the easier it will be to use an edge finder to touch off against the sides.

The deeper the bottom plate, the more evenly the vice jaws will grip it. The broader the bottom plate, the more stable the grip. The longer the bottom plate, the more area of grip for the vice jaws.

Material

- Aluminium plate or block machines easily and is relatively soft so that it causes less wear and tear if the cutter bites into the plate.

Tools

- Use a large-diameter end mill, long enough to take a finishing cut down the whole of the vertical face of each individual plate.
- Use a large-diameter fly cutter or face mill to surface the top of the upper plate.

Speeds and Feeds

For aluminium, the speeds and feed rates are shown in Table 8-1.

Method

- Cut the upper plate a little oversize (up to 3mm or 1/8in larger on its width and on its breadth).

Table 8-1

Tool	Spindle speed (rpm)	Feed rate: roughing	Feed rate: finishing
10mm (⅜in) 4-fute end mill	1,800 to 3,000	1,500mm/min (60in/min)	300mm/min (12in/min)
50mm 3-fute face mill	2,500	2,000mm/min (80in/min)	500mm/min (20in/min)
50mm single-point fly cutter	2,000	500mm/min (20in/min)	100mm/min (4in/min)
3mm (⅛in) HSS twist drill	Up to 6,000		300mm/min (12in/min)
3mm (⅛in) reamer	Up to 2,000		100mm/min (4in/min)

Note: For a 3mm (⅛in) reamer, peck-ream in small intervals with lots of lubricant, and clean the reamer flutes if necessary. Reamers should always turn in the direction of cutting (usually the same direction as a twist drill), even when being retracted.

- Clamp the plate on the bed or hold it in the vice, taking care to set the top of the plate horizontal a little above the top of the jaws, because this allows space under the plate for the reamer to pass through as well as allowing a cutter to take a skim cut across the top of the plate.
- Touch off at the approximate centre of the plate, then drill the four holes and countersink the two larger holes. All four holes can be centre-drilled or spotted; then all four can be drilled with a 2.9mm drill (drill size to allow for reaming the two 3mm-diameter holes). The two larger holes can be drilled to final size and then countersunk. Then the two smaller holes can be reamed to final size. These holes are for 3mm dowels to align the top and bottom plates, so they need an accurate finish.
- Stand the smaller bottom plate vertically in the vice and mill the edges square. Finish one side, use an engineer’s square to set that edge vertical so that the adjacent edge will be milled

horizontal, and then reference opposite edges against the inner base of the vice to finish the other two sides.

- Grip the bottom plate horizontally in the vice, just as you did with the top plate, and touch off at the approximate centre of the plate.
- Drill all four holes, leaving the dowel holes slightly undersize for reaming.
- Drill tapping-size holes for the bolts.
- Ream the dowel holes to size.
- Assemble the plates, inserting the 3mm dowels then the bolts.
- Then put the assembled plates in the vice, in the operating position, with the larger plate facing upwards. If the tops of your vice jaws hold the plate clear of the main body of the vice, sit the underside of the plate right down on to the jaw tops. If the tops of the jaws sit flush with the vice body, use two little spacers on the top of the jaws to allow an end mill to cut right round the sides of the plate without cutting into the vice. Two same-size drills would do here, or two pieces of sheet metal.
- Grip the bottom plate securely, then take a facing cut right around the periphery of the top plate.
- Finally, sit the underside of the top plate right down on to the tops of the vice jaws, and skim the top surface of the plate. The best tool for this is a fly cutter or a face mill, but it can be done with multiple passes of a smaller-diameter end mill.
- Mark one edge so that you will remember to place that edge on the left. That way, when you use the plate as a fixture, you will be able to touch off against that edge and against the front edge to set the origin for the plate.

In use, the workpiece will be secured in a known position in relation to the origin of the plate by drilling holes and adding pins and clamps, or something similar. That way, you will be able to use appropriate offsets so that the Controlled Point is in a known position in relation to the workpiece, because you know where the workpiece is in relation to the origin of the plate.

If, for example, you know that a suitable origin on the workpiece is located 10mm to the right and 15mm from the front edge of the fixture plate, you can touch off at the sides of the fixture plate, but with offsets of X−10 and Y−15, less half the diameter of the tip of any probe you use to touch off.

Be prepared to drill holes in the top plate as required for each workpiece, for dowels and for clamps, but take a note of which holes are to be used for each workpiece. The plate will eventually resemble Swiss cheese, but you will find it very useful. When the top surface becomes grooved and marked, you can always skim it again to provide a fresh flat surface.

The parts are separated from the block using a thin, large-diameter slitting saw.

Making multiple parts from a single sheet or block is a lot like machining the same feature in different locations on one workpiece, as shown in Chapter 7, and the same methods can be used.

Instead of machining everything from one sheet of material, there could be several separate fixtures arranged so that a set of different workpieces can be machined by one program. As long as the position of each fixture is known accurately, this should work just fine. You might, for example, set up fixtures to make the parts for a single mechanism, like a set of clock parts, so that instead of machining several copies of each item and holding a stock of multiple copies of each part, you could machine complete sets of parts at one time by using a set of fixtures. It all depends on what you want to do.

While it is easy enough to machine a set of parts of identical thickness from a single sheet of material, this will not work if the material to be used for each object is of a different size and shape.

The set of parts shown in Fig. 8-4 are of different sizes and thicknesses, but could be machined by one program if they were set up in different fixtures, provided the fixtures were all at known positions. This could be done by measuring the position of each fixture, by using gauge blocks to space the fixtures accurately in

relation to each other or by careful positioning of the fixtures on one plate.

Fig. 8-4 Steam engine castings that could be mounted on a single fixture plate and machined as a batch.

However, it is unlikely that the **G92** command will be up to this job. **G92** redefines the current point, but that has its limitations and another approach is often easier.

If there are several fixtures in use simultaneously, it is possible to set up a separate coordinate system for each fixture. That way, when machining is to take place using a particular fixture, the Controlled Point can be made to use the coordinate system for that fixture. That coordinate system will have its own associated work origin that should be set to coincide with the origin of the fixture.

In practice, coordinate systems are defined as sets of offsets, each offset being from the absolute machine coordinates. So each fixture would have an associated set of offsets representing the position of its origin in relation to the absolute origin of the absolute machine coordinate system.

GAUGE BLOCKS

Gauge blocks are precision-made blocks, usually of steel or ceramic material. They can be obtained individually, but are normally supplied in sets with sizes chosen so that stacking the blocks can make any size within the range of the set, to within a very close tolerance. Tolerances vary between +/−0.05µm (+/−0.002µin) for referencegrade blocks and +0.25µm/−0.15µm ((+0.01µin/−0.006µin) for workshop-grade blocks. Accuracy demands care in storage and use, and blocks should never be left stacked together.

The **G54** to **G59** commands define offsets that determine the position of coordinate systems for fixtures (and the corresponding workpieces) in relation to the absolute machine coordinates. Assigning offsets to fixtures means they effectively have their own coordinate systems, with their own origins located away from the absolute machine origin (0, 0).

When the ‘normal’ work origin is set by touching off, that is the default **G54** origin, but other origins can be defined in much the same way.

Offsets are stored internally by the CNC software, but can be saved to a file so that they can be used the next time the CNC software runs. Values of the fixture offsets are held in a table, in groups of six values for each fixture. Each group holds offsets for X, Y, Z, A, B and C axes, in order. The syntax of the commands for using offsets is different for LinuxCNC and Mach3. Whereas LinuxCNC has up to 9 sets of offsets, Mach3 allows up to 254 sets of offsets.

LinuxCnC

Table 8-2

Fixture number	Command to set offsets	Axes for each fixture			
X	Y	Z	A	B	C
1	G54				
2	G55				
3	G56				
4	G57				
5	G58				
6	G59				
7	G59.1				
8	G59.2				
9	G59.3				

Note: The syntax of the **G59.1** to **G59.3** commands applies only within LinuxCNC.

Mach 3

There are 254 available offsets. For convenience, the first six are accessible using commands **G54** to **G59**. All offsets can be accessed using **G59 P~** where the P parameter identifies the offset number (1 to 254).

Table 8-3

Fixture number	Command to set offsets	Axes for each fixture					
		X	Y	Z	A	B	C
1	G54 or G59 P1						
2	G55 or G59 P2						
3	G56 or G59 P3						
4	G57 or G59 P4						
5	G58 or G59 P5						
6	G59 or G59 P6						
7	G59 P7						
8	G59 P8						
And so on, until…							

254 G59 P254

Note: The syntax G59 P~ applies only within Mach3.

The offsets for fixtures are entered into the table using the **G10 L2** command.

The **G10** command takes two forms: **G10 L1** and **G10 L2**.

G10 L1 assigns values to another table, for tool offsets (see later), and the fixture values are set using **G10 L2**.

The general form of the **G10 L2** command is **G10 L2 P~ X~ Y~ Z~ A~ B~ C~** where P~ is the fixture number (1 to 9 for LinuxCNC; 1 to 254 for Mach3), so P1 sets the offsets for fixture 1, P2 sets offsets for fixture 2 and so on. In practice, this is best done by touching off at each of the fixtures.

G10 L2 defines offsets, but these are from absolute machine coordinates, not from the current work origin, so there is some arithmetic to do. Because it is not always easy to find the absolute machine coordinates, this makes using the **G10 L2** command difficult.

LinuxCNC contains a more useful form of the command to avoid this.

Now using the **G54–59** commands will switch the coordinate systems to suit the fixtures or machining positions being used.

LinuxCNC only (and only from version 2.5 onwards):

Take the Controlled Point to the position at which an origin is to be set for a coordinate system, then use* G10 L20 P~ X~ Y~ Z~ *instead of* G10 L2*. The* L20 *form of the command calculates the offsets needed to make the current point have the values given in the command, then puts those offsets into the offset table.

So; taking the Controlled Point to a position, then using

G10 L20 P2 X0 Y0 Z0

will make the software calculate the offsets required to make that point (0, 0, 0) and store those offsets in coordinate

system 2 (that is, the* G55 *system).

G10 L20 P3 X10 Y0 Z25 ***will put the offsets needed to make the current point (10, 0, 25) in coordinate system 3 (*G56*) and so on.***

Offsets are defined in relation to the absolute machine coordinates, but it is useful to have a consistent reference for human convenience, and that might be the G54 origin (which is itself defined by offsets from the absolute machine coordinates. just like the other G55–59 coordinate systems).

In practice, go to a convenient point on a fixture, touch off, setting G54 P1 to (0, 0, 0), then either:

***move the Controlled Point to a suitable reference point on each fixture in turn, and define that as (0, 0, ~) by touching off* G55–59 *as appropriate, or by using* G10 L20 P~ X0 Y0 Z~**

or

remaining at the* G54 *work origin, for each of the fixture reference points in turn, enter the coordinates that* G54 *work origin would have if the fixture reference point was (0, 0). Enter those coordinates by touching off or by using the* G10 L20 P2 X~ Y~ Z~ *command to make the software calculate and enter the* G55 *offsets from absolute machine coordinates into the offsets table.

G54 followed by **G0 X0 Y0 Z0** will take the Controlled Point to that first work origin for the fixtures. **G55** followed by **G0 X0 Y0 Z0** will take it to the second fixture, **G56 G0 X0 Y0 Z0** the third and so on.

Early in a program that will make use of fixtures and offsets, make sure you are in **G54**, at a known position (normally the origin), then put a list of **G10 L20** commands to assign the offsets for each fixture or each machining position on a fixture plate. Later in the program, when you need to carry out machining at a fixture, use the corresponding **G54** to **G59.3** command to switch to the coordinate system for that fixture.

At the end of the program, switch back to the offset system you will use for the next session. **G54** is usually the coordinate system set by touching off to get the first work origin, so that is very often the system you will use unless you have a specific reason to do otherwise. You may also wish to erase the other offsets, and this is most conveniently done using the **G10 L2** command to put offsets of 0 into X, Y and Z for each coordinate system except **G54**.

A typical program using fixtures might have a structure like this:

- Comments
- Initial definitions

Mach3: subroutine definitions go after the end of the program)

- Define subroutines
- Define fixture offsets using **G10 L20 P~ X~ Y~ Z~ A~ B~ C~**

When you need to machine the work in a particular fixture:

- Use **G54–59** as appropriate.
- Use the commands to carry out machining at that fixture.

When you need to machine the work in another fixture:

- Use **G54–59** as appropriate (which automatically replaces the previous fixture offsets).

When you have finished using the fixtures:

- Put 0 into all the offsets in all the coordinate systems **G55 to G59**.
- Use **G54** to return to the ‘normal’ default work origin.

The benefit of this approach is that the fixture offset values are only defined in one place: at the start of the program. If fixture positions change, only the values in that section need to be changed. Life being as it is, fixture positions will inevitably change, even if only by a small amount, every time the fixtures are set up, so easy access to the commands to define the new positions is important. It is particularly important that you do not have to hunt through the program to change values in several places to alter the fixture offsets, as this will inevitably lead to errors.

A Further Word on G92

G92 is an overall global offset that can apply over and above any fixture value. This means you do need to be careful that you know what offsets apply at each stage of your program. Normally, if you are using fixture offsets you would not need **G92** as well, but it is not impossible to imagine a situation where there are two complete sets of fixtures, each held on their own subplates, and **G92** is used as a global offset to locate the second plate in relation to the first plate, with **G54** still being the main reference point at the work origin for a plate, and **G55** to **G59** offsets for the individual fixtures being from the **G54** origin of that whole plate. Once that was done, individual fixture offsets would be used to machine the workpieces. On each plate, the relative positions of the fixtures will not change from one set-up to the next, and it is only the relative positions of the plates that may change. The **G92** offset may be different each time the plate is fastened to the mill table, but the coordinate system offsets **G54** to **G59** will remain the same within each plate.

G92 remains in effect throughout the whole program, because it is a global offset, and it is wise to cancel it using **G92.1** at the end of the program, otherwise it will persist.

TAKE CARE WITH G92

Using G92 when other offsets exist requires a very clear head.

The Mach3 manual rightly says ‘You are strongly advised not to use this legacy feature on any axis where there is another offset applied.’

As with all complex features, G92 is powerful but can have unfortunate consequences so keep your wits about you. If you are not sure if G92 is in effect, use MDI mode to type G92.1. If the DRO readings change, G92 was in effect.

CAM Programs and Work Offsets

Most CAM programs do not take account of sets of fixtures and **G54–G59** coordinate systems. Instead, use a CAM program to create a set of programs, one for each fixture. These become subprograms that you can link within a larger program, with appropriate **G54–G59** commands between each of the subprograms.

Using a Single Sheet or Block of Material

In this case there are no actual fixtures, but the concept of virtual fixtures is useful here. A single sheet or block can be treated as a set of fixtures mounted on a plate and, although there are no physically separate fixtures, the fixture offsets represent positions on the material where machining takes place.

With a set of separate fixtures, there will inevitably be minor variations in position every time they are set up, so new offsets will be needed each time.

With individual fixtures already mounted on a common plate, the position of each fixture will not change in relation to the other fixtures, so the only thing that will change is the position of the plate itself, and that can be dealt with by touching off (**G54–G59** as appropriate) at a suitable reference position on the plate. In a similar way, using a single sheet of material will mean the individual offsets of the cutting positions from the origin of the sheet will not change; it

is only the overall position of the sheet on the machine table that may change between set-ups. Because of that, **G92** can be used to define the position of the origin of the sheet, and the **G54–G59.3** values for the cutting positions within the sheet will be unchanged.

Using a CAM Program with a Single Sheet of Material

CAM programs are well able to cope with machining individual workpieces from a single sheet of material. In fact, they excel at this. Simply draw the workpieces on a sheet in any position or orientation, and the CAM program will output a single complete program to machine them all in one session.

More fixtures

G54–G59.3 are adequate for up to nine fixtures in LinuxCNC or 254 for Mach3, so LinuxCNC can be short of available fixture offsets in some circumstances. If more fixtures are needed, this poses a problem with an inconvenient solution in software.

With a little foresight, it is possible to set up another section of program that redefines the offsets for the next set of fixtures so that once the initial set of offsets have been used for the first set of fixtures, and all the machining has been done at those fixture positions, the next set of offsets can be defined and used for the next set of fixtures. This needs to be done with care.

However, instead of doing everything in software, some hardware aids can be used. Fig. 8-5 shows a fixture mounted on a rotary table. The clock base on the table requires holes for three feet spaced 120 degrees apart, so the table is simply rotated to bring the next machining location into position. There is no need for alternative coordinate systems or offsets. Although this arrangement of three holes could easily be machined with the work clamped directly to the mill table, the spacing of these holes is too large for the cross-slide

movement, and the rotary table provides a way of reaching all three holes.

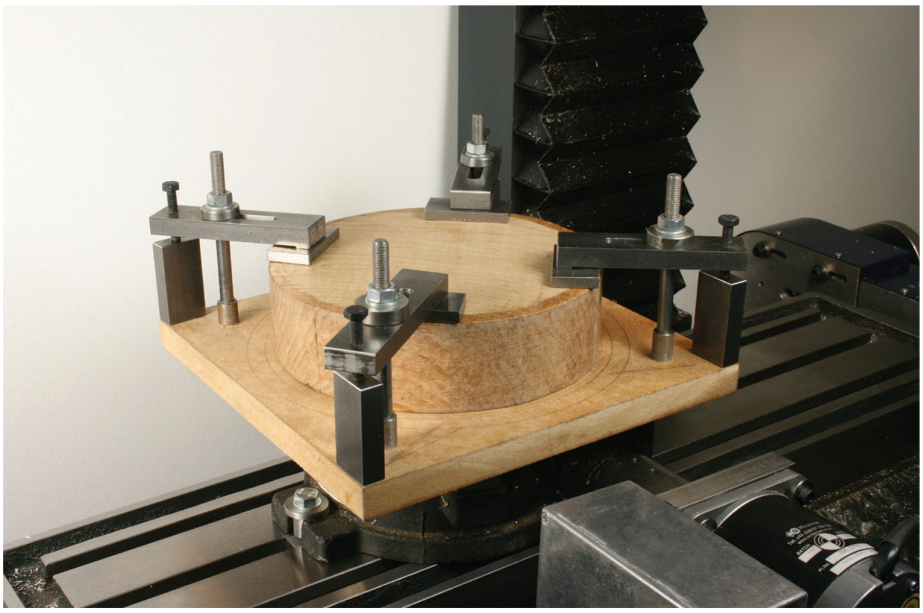


Fig. 8-5 Mounting this clock base on a fixture on a rotary table (fourth axis) allows a small mill to machine positions it could not otherwise reach.

A rotary table can be mounted on the mill, as shown, with its surface horizontal or, if machining is to take place around the edges of the workpiece, the rotary table can be mounted so that the plane of its table surface is vertical.

Using the same principle, several fixtures might be mounted on a rotary table so that each movement of the rotary axis brings another fixture into position. This works best when the fixtures are arranged on a single plate; otherwise, it becomes difficult to set the fixtures in position on the table.

With this kind of arrangement, programming the rotations cannot be done using a CAM program, unless it has been purposely designed with this in mind. Instead, the CAM program can be used to generate the program required for the fixture on one face. The resulting code is then copied and pasted so that there is an identical block of code for each face. Fourth-axis rotation commands can then

be placed between those blocks of code to make a complete program. This is a good example of an instance where some knowledge of G code is a big advantage.

NESTING PARTS

One advantage of machining several parts from a single piece of material is that the material only needs to be set up on the mill once for all those parts to be produced. Another advantage is that there may be less overall waste. Machining a 50mm (2in) circular part from a square piece of material is sensible if the corners are used as clamping points, but there is considerable waste, as shown in Fig. 8-6. Machining several circles from a single larger sheet means there is more area that is not covered by clamps, and less required between parts, so other parts could be machined from those areas that would otherwise be scrap, as shown in Fig. 8-7.

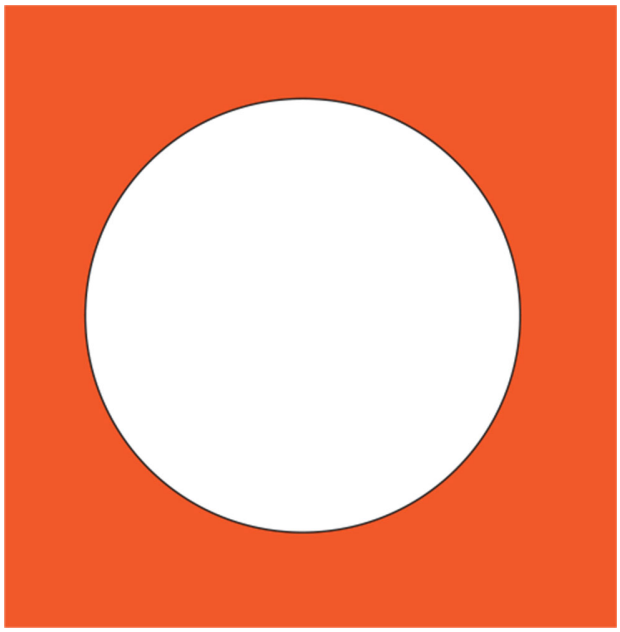


Fig. 8-6 Cutting a single circle from a square of material produces waste.

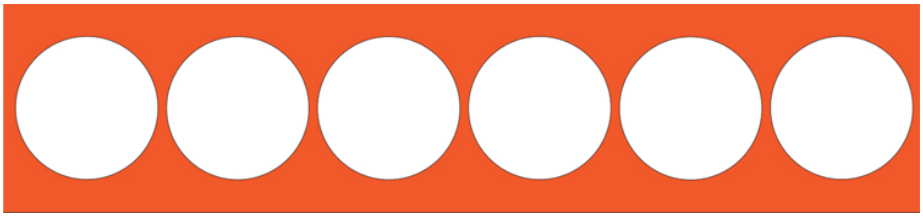


Fig. 8-7 Cutting several parts from one sheet of material can help reduce waste.

Placing several separate parts to be machined from a single piece of material is termed ‘nesting’, and the object of nesting parts is to minimize waste as well as reducing the time spent setting up material.

Nesting needs to be done by bearing in mind the size of the cutters that will be used so that if the periphery of a shape is being machined, it does not cut into the area required for other parts. There is a skill in nesting, and this is one of those situations where a good software program can help, but the human eye and brain can sometimes find a better way of nesting shapes than the computer.

Nesting shapes efficiently often requires that parts are rotated from their original design orientation; so a part designed in what seemed like an obvious upright position may be efficiently nested by rotating it and moving it in relation to other parts. This means the code must be rewritten because the Controlled Point must move in an entirely different path when machining that part. This is best done using a CAM program, especially where many parts are being nested. The nesting function in VCarve Pro, for example, can move and rotate parts while taking account of the space that needs to be left between parts to allow a cutter to machine their peripheries. Fig. 8-8 shows an example of nested parts.

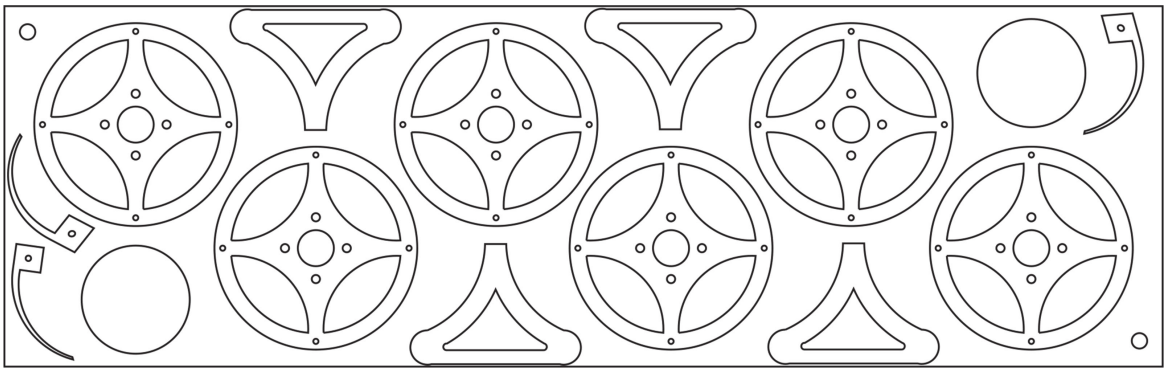


Fig. 8-8 A set of nested parts cut from a single sheet of material can substantially reduce

Project 8.2

Cable Chain

A cable chain is a protective tunnel for cables that have to move backwards and forwards as an axis of the CNC machine moves. The chain guides the cables and prevents them from kinking or from bending too sharply. Figs 8-9 to 8-11 show a cable chain in various configurations as it rolls up or down.

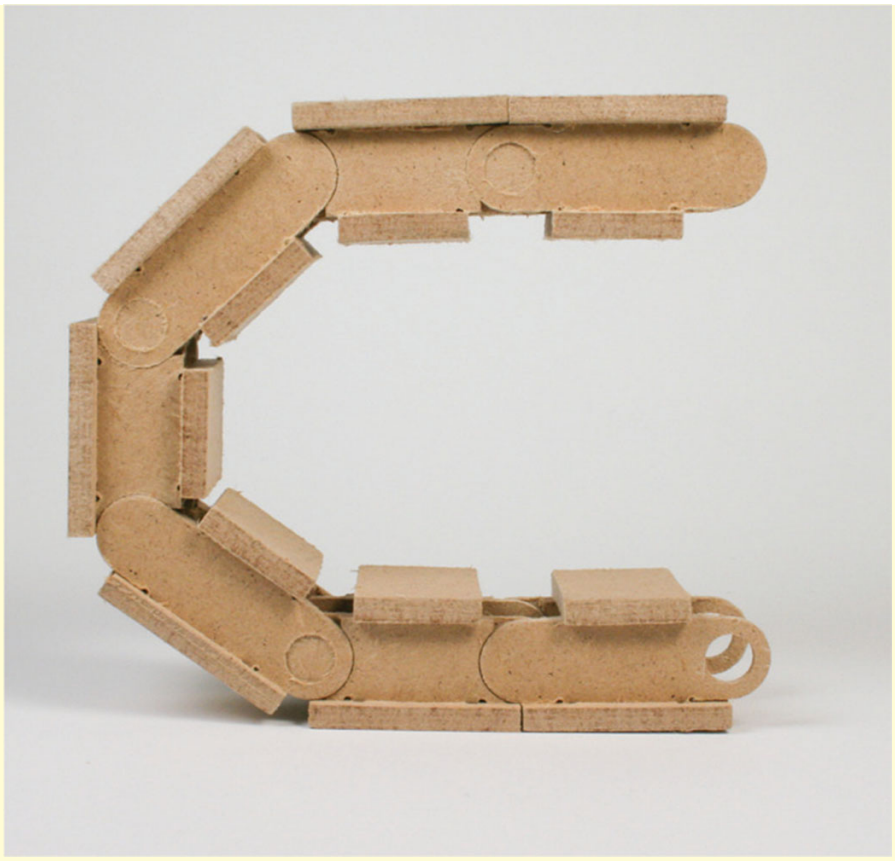


Fig. 8-9 A section of cable chain in a curved formation.

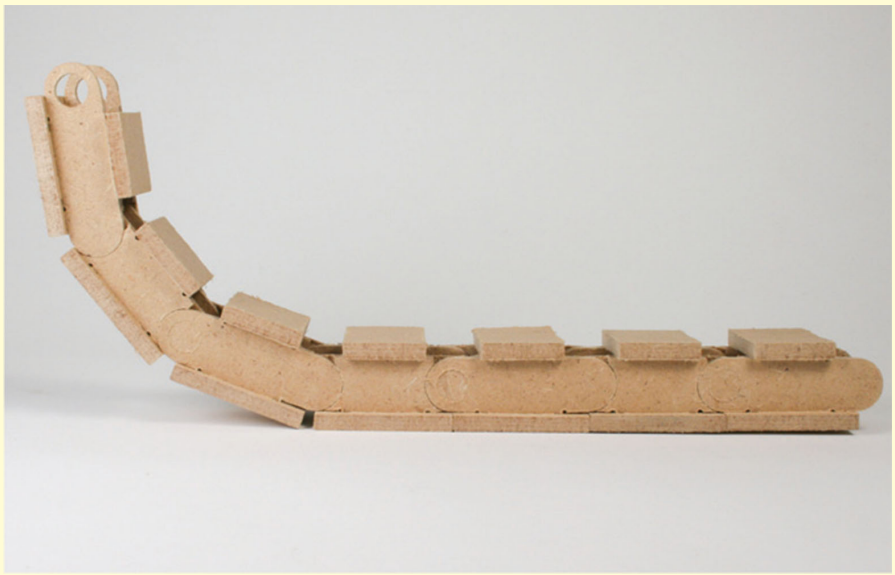


Fig. 8-10 A section of cable chain partially curved.

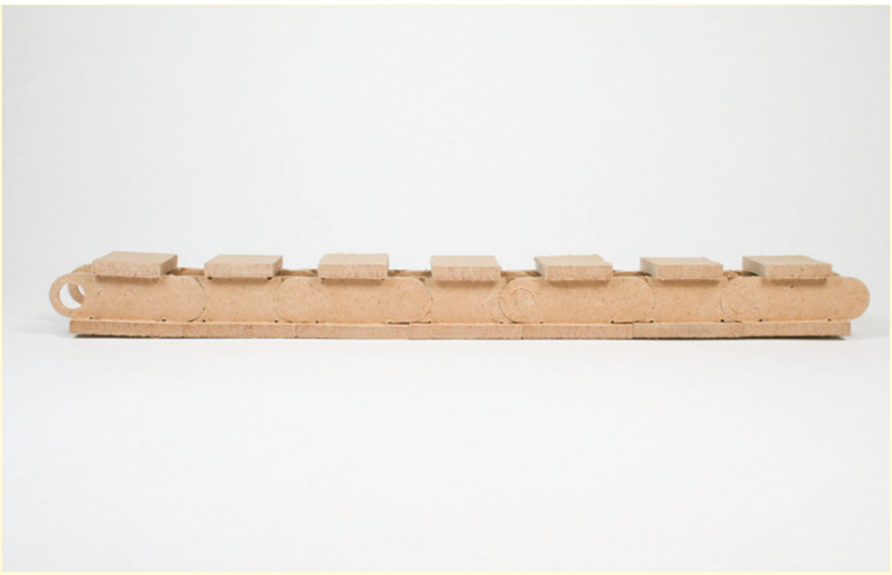


Fig. 8-11 A section of cable chain lying flat.

Individual cable chain links are made from a small number of parts, and the chain in this project uses just four parts for every link. Because a complete chain uses a lot of links, machining the parts for a chain makes good use of a CNC milling machine.

The drawings shown in Figs 8-12 to 8-15 show the individual parts of the chain and Fig. 8-16 shows a single assembled link. The dimensions can be modified to suit the materials and the width of the channel required inside the chain to suit your cables.

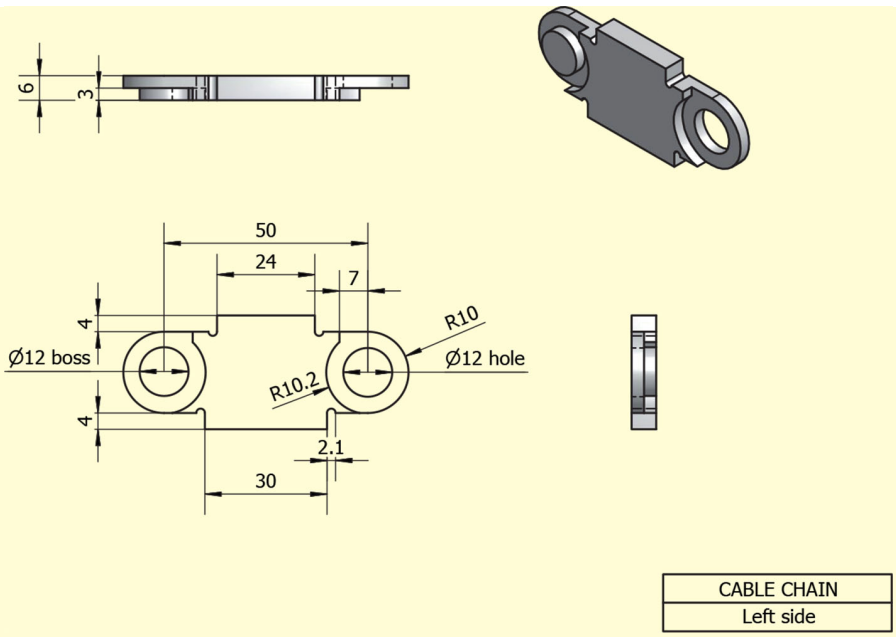


Fig. 8-12 Cable chain left side link.

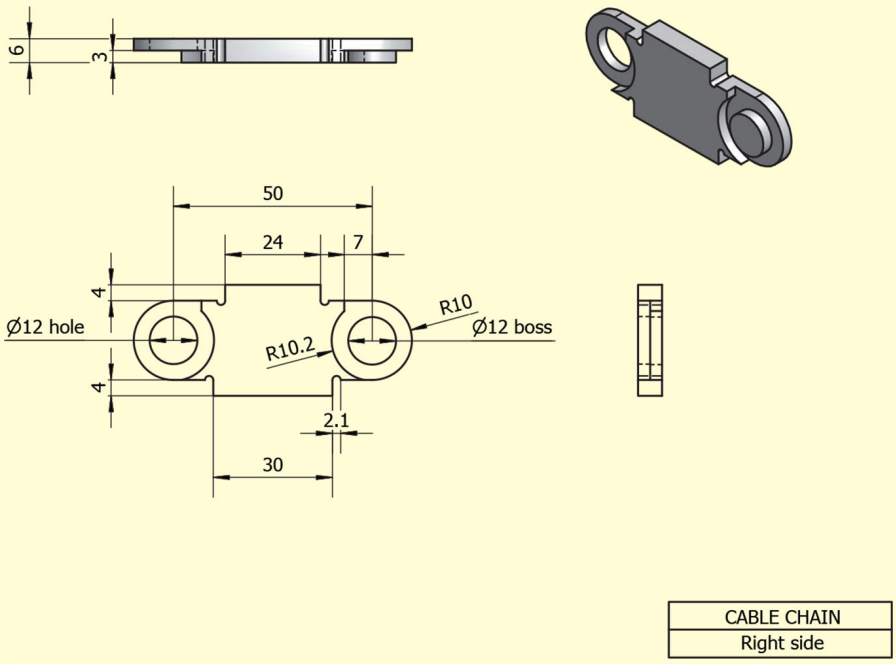


Fig. 8-13 Cable chain right side link.

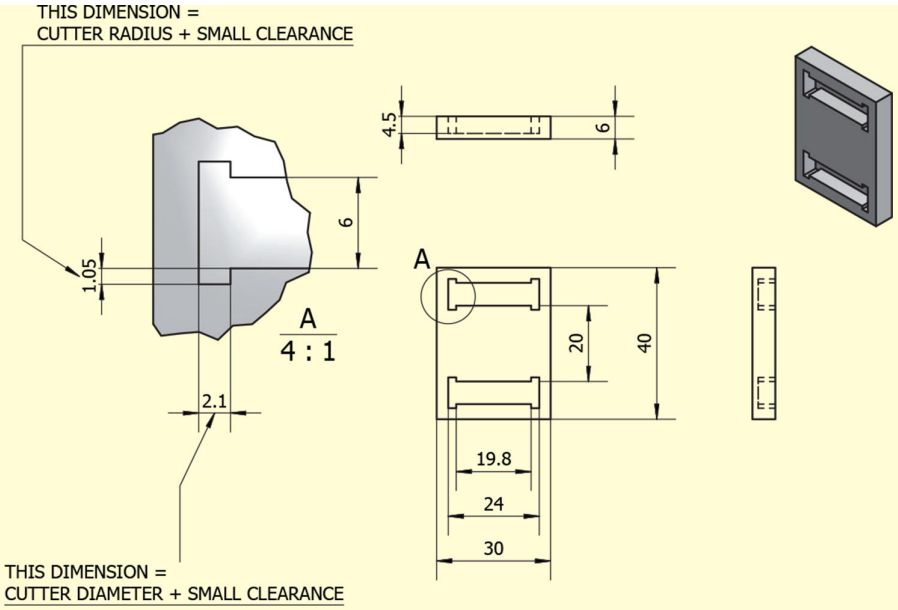
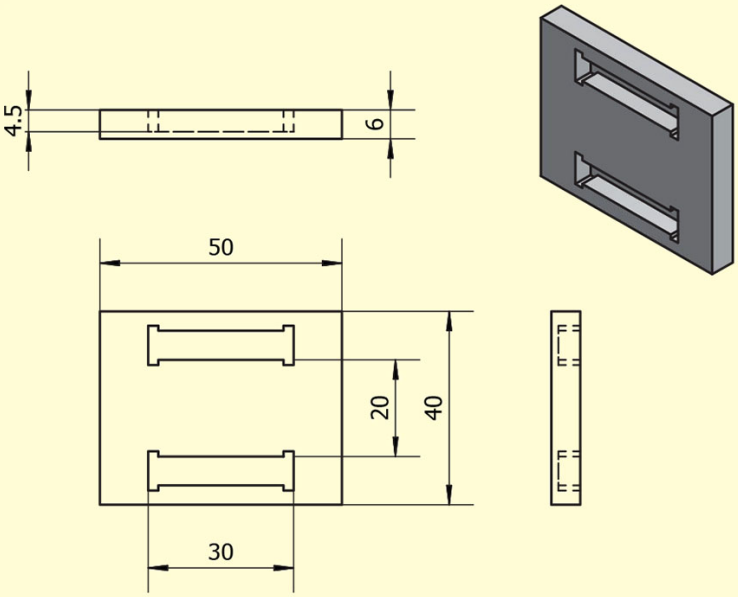


Fig. 8-14 Cable chain top plate.



For details of additional slot dimensions
refer to Top Plate drawing

CABLE CHAIN
Bottom plate

Fig. 8-15 Cable chain bottom plate.

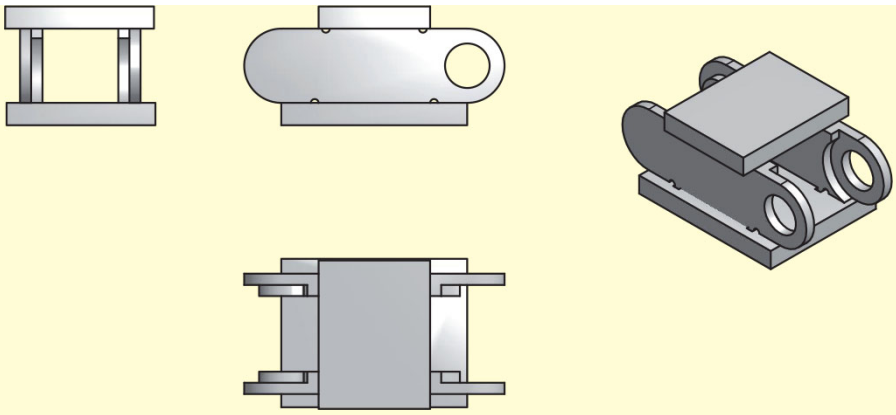


Fig. 8-16 Assembled link.

Fig. 8-17 shows how the sides link together, and careful study will show that links are assembled with the sides alternated to fit one link to the next.

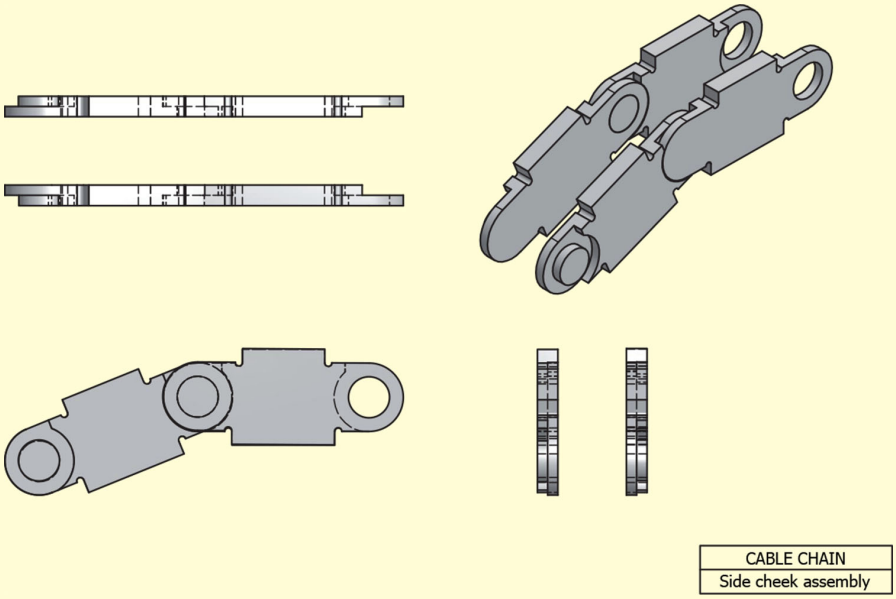


Fig. 8-17 Side cheek assembly.

Design Notes

The width of the slots should be altered to give a close fit on the material. The drawings are based on a material thickness of 6mm, but many materials that are nominally 6mm are not 6.000mm

thick, so some of the drawing sizes will have to be altered to suit. The depth of the recessed faces at the end of each side is exactly half the actual thickness of the material, so measure the material carefully before setting the depth of cut.

In the top and bottom plates, the distance between the inner sides of the slots defines the cable-carrying capacity and must be the same for the top and bottom plates. The dog-bone shape of the slots is to avoid rounded ends on a plain slot, which would not suit the squared ends of the projecting tabs on the sides that need to sit in slots with square end faces. Rounded ends as produced by an end mill or slot drill would not provide a sufficiently accurate end location. The dog-bone shape moves the curved ends of the slot to a position where it will have no effect on the end faces in the slot; Fig. 8-18 shows the path of the cutter as it forms those clearance pockets. Taking the milling cutter to the sides, by slightly more than the radius of the cutter, allows the usable parts of the ends of the slots to be flat and square to the length of the slot so that the projecting tabs at the tops and bottoms of the sides fit snugly within the slots, end to end.

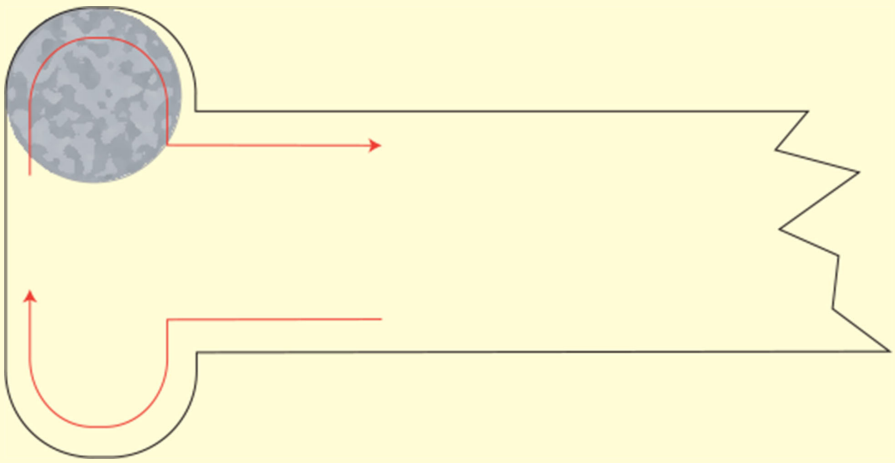


Fig. 8-18 The cutter path required to produce dog-bone ends.

Each side piece has four small semicircular grooves where the edges turn upwards through 90 degrees; these are to remove any

obstructions at the corners, to allow the top and bottom plates to sit against those edges. This is a similar principle to the dog-bone ends to the slots in the top and bottom plates. These semicircular grooves are a little larger than the diameter of the cutter so that the cutter has room to move in a tiny circle, rather than being fed straight into the groove. A little clearance here helps prevent the cutter flutes being clogged.

The inner radius on each side is larger than the radius on the outer edge (R10.2 compared to R10) to provide clearance where the sides fit together along the chain.

Material

- A light, hard plastic that can be glued is ideal.
- MDF or similar composite board will work in an oil-free workshop.

Tools

- A small end mill or slot drill, or a small-diameter router cutter, preferably centre-cutting. The diameter must be smaller than the smallest diameter of the curve (2.1mm).

Speeds and Feeds

- Speeds and feeds depend on the materials, as always.

Method

A test piece should be used to help determine fits for the material being used. Parts should fit snugly but the hinges should operate freely.

Parts can be nested, and the choice is either batches of identical parts or batches of the four component parts at a time. Fig. 8-19 shows one suggested layout for a batch of component parts for two links. Generate the G code for each part individually.

Use work offsets **G55** to **G59.3** (or **G59 P1** to **P254**) to define the work origins for each part in the layout. Then use one of these methods:

- Use MDI mode to select a work offset and then run a program to machine a single part at that location.
- Edit the G code for a batch of parts in one program, with appropriate **G59** commands between each section of code, then run this as a single program to machine a whole batch of parts at a time.

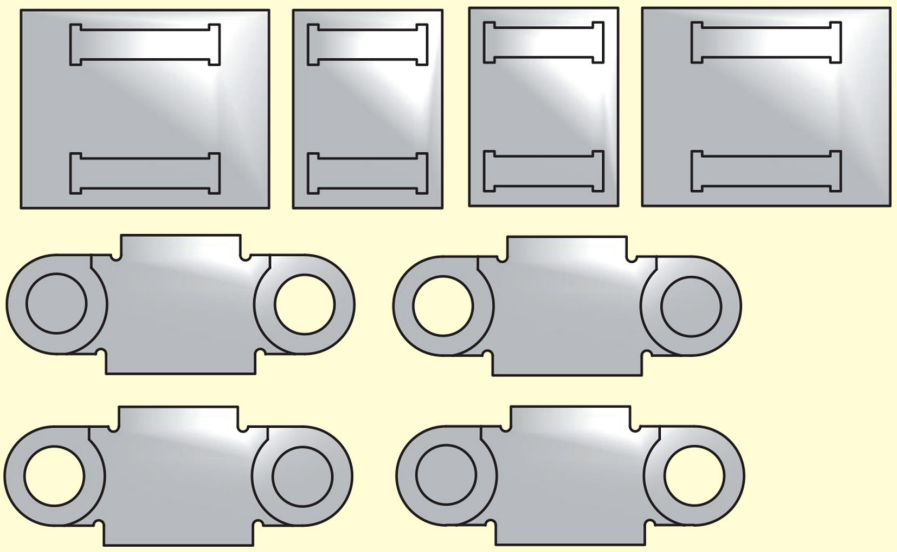


Fig. 8-19 One possible layout of nested parts to produce cable chain links.

An automatic toolchanger fitted to a Tormach PCNC 770. (Photo: Tormach LLC.)

9 Tool Tables, Cutter Compensation and Tool Length Offsets

In this chapter you will learn about:

- how to compensate for tools of different lengths and different diameters by using tool tables.

To be able to do a successful manual tool change during a program, you really need to be using tools fixed in a holder that mounts into the spindle in a repeatable way, so that each tool always projects a known distance from the spindle every time it is mounted.

If you are using collets, they will not hold tools with a repeatable projection, so a better way to cope with this is to split your program into smaller individual programs. If a task is done in stages, with a tool change between each stage, make each of those stages of machining a separate program. The tool height can then be set in the usual way and touched off before running each program. You will not need tool tables for that approach.

The information below about tool length offsets is of use in situations where tools are held repeatably. The information about tool diameters is useful for any tool, no matter how it is held in the spindle, and is not affected by the tool length or tool changing methods. Tool diameter information is used to compensate for the difference between the path of the Controlled Point and the position of the edge cut by the tool.

TOOL TABLES

The **tool table** is a list of tools and their characteristics, like the tool length and the tool diameter. The tool table is a file stored independently of the CNC program, so it is possible to work with the same tool information after shutdown and restart. This can be very useful.

Mach3 can store information about 256 tools in the table. LinuxCNC can store information about a maximum of 56 tools in the tool table.

Each tool is referred to using a P number, which is the pocket number where the tool is held in a mechanical tool changer. On a mill, the most useful entries for each tool will be the tool length offset (Z) and the tool diameter (D). For a programmed toolpath, putting a longer or shorter tool in the spindle will make the cut deeper or shallower than intended; changing the diameter of the cutter will machine edges too far to the right or the left.

G49 cancels any tool offsets currently being applied. **G43 H~** applies the tool length offset for tool number ~ in the tool table.

LinuxCNC only: after an M6 tool change (for example, T2 M6), a G43 command applies the offset for that tool number from the tool table.

Mach3: In Mach3, the useful tool table entries can be made using the Offsets screen, or in G Code, or by editing the tool table file in a text editor. By far the best way is to use the Offsets screen.

LinuxCNC: In LinuxCNC, File > Edit tool table starts an editor that allows entries to be made into the tool table.

Tool length offsets should be positive numbers, so a little organization is necessary when a job will involve more than one tool (Fig. 9-1).

The longest tool should be considered as having 0 offset. This ‘tool’ could be a reference rod, simply chosen to have a length greater than any of your tools. Each shorter tool will have a positive offset that is a distance from the end of the current cutter to the end of the reference bar. Imagine touching off the reference rod (or longest tool) and setting that to Z0. The end of a shorter tool will be located at a higher Z value and it is that value that is entered into the tool table.

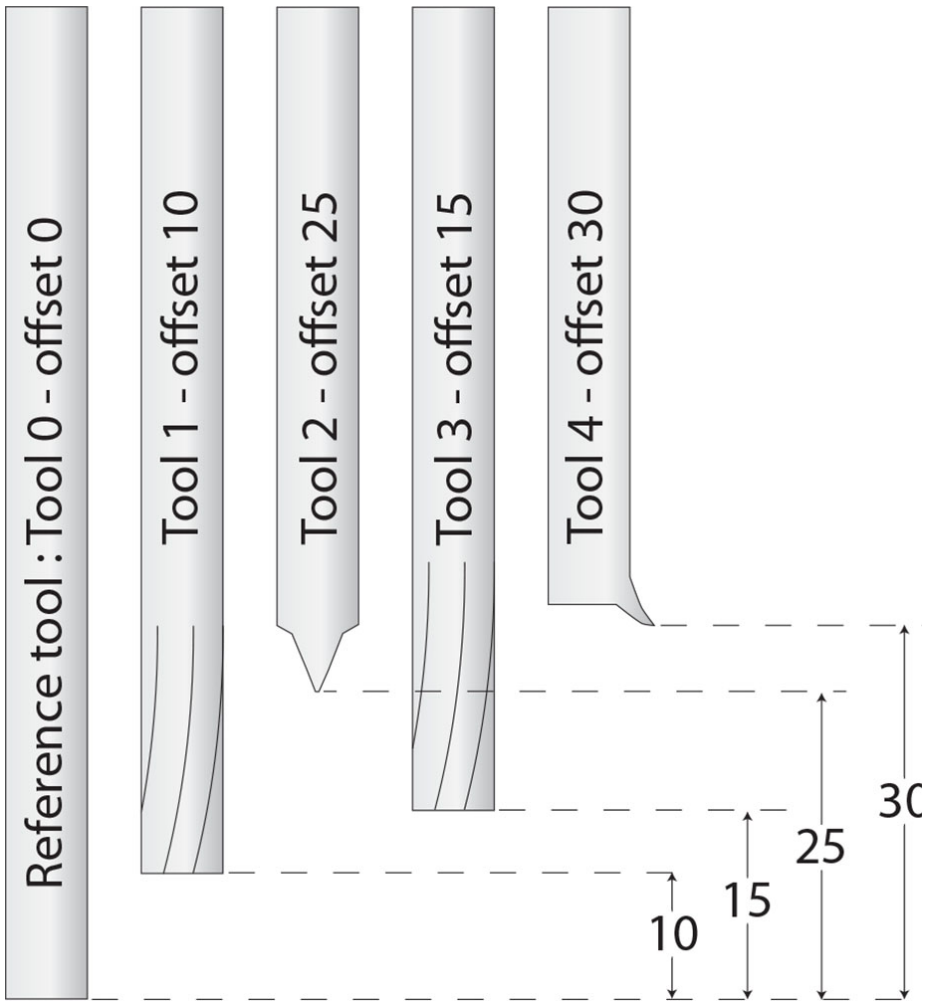


Fig. 9-1 Cutter offsets from a reference tool length.

Another way to do this is to lower the spindle with the reference rod (or tool) on to a Z height gauge and touch off Z there. That makes the Z height 0 for that rod (or tool). Then lower the next tool on to the Z height gauge and note the height. Ignoring the negative sign will give the offset, because the offset is the extra height of the Controlled Point above where it was for the reference tool.

Repeat this process for each of the tools.

Tool length offsets can be entered into the tool table by using the **G10 L1** command **G10 L1 P~ Z~** where P is followed by the tool number and Z by the tool height offset.

G10 L1 P2 Z14 makes an entry for tool 2, entering 14 as its tool length offset.

The tool diameter cannot be entered this way.

Tool Length Offsets in LinuxCNC

In LinuxCNC, tool length offsets can be entered using the Touch Off button, but only following a **T~ M6** command.

In MDI mode, type **T1 M6** to select tool 1 (the **T1** command) and make a tool change (the **M6** command). This should produce a window telling you to mount tool 1. Click OK to dismiss that window.

NO TOOLCHANGE WINDOW

In LinuxCNC, the Toolchange window may not appear if you have previously issued a similar tool change command and the software believes the tool you have requested is already in the spindle. This is not normally a problem.

Make that tool change. Then go to the main screen, select the Z axis, jog and touch off in the usual way, using a probe or a roller under the tool, or your favourite method. Instead of defining a **G54** offset, use the menu to choose Tool Offset.

Go back to MDI mode and type **G43** to implement that tool change. The Controlled Point will not move, but the DRO will now read correctly for the tool currently in the spindle, taking account of the tool offset.

CHANGING TOOLS

The **T~** command prepares for a tool change but it does not take place until the **M6** command is issued, so **T3 M6** requests that tool 3 be mounted in the spindle. Note that it is acceptable for the **T~** and **M** commands to be on the same line, in which case the order does not matter. **T3 M6** is as effective as **M6 T3**.

How the tool change is done, physically, depends on your hardware and on how your CNC software expects to handle tool changes.

Implementing a Tool Change in Mach3

The behaviour of Mach3 when it meets the M6 tool change command is set using the Configure>General Config… menu. There are three options here.

The most basic option is to make Mach3 ignore tool change requests. This is fine if there are no tool changes.

For manual changes, the behaviour can be set to ‘Stop spindle. Wait for Cycle Start’.

Finally, Mach3 can be set up for automatic tool changers. The way in which this is done depends on your tool changer; commercial tool changers should be supplied with a bit of code (a *macro*) that Mach3 can run when it uses that particular tool changer.

For automatic changers, the **M6** command makes Mach3 execute two macros, one after the other, pausing only to wait for the Cycle Start button to be pressed between macros. These two macros, M6Start and M6End, must be custom-written in Visual Basic

by the user to suit the changer being used, so this is not a trivial undertaking.

Implementing a Tool Change in LinuxCNC

The **M6** command allows for the possibility that the Controlled Point moves to a preset position to allow you to do a tool change, and that facility helps with tool changing.

The **G28** and **G30** commands are useful here, but note that they are implemented differently in LinuxCNC and Mach3. Note also that G28 and G30, and their variations, should not be used if tool diameter compensation is in effect.

G28 AND G30 IMPLEMENTATIONS

The G28 and G30 commands, and their variants, make use of two groups of system parameters. LinuxCNC uses variants G28.1 and G30.1 to store values in those parameters; Mach3 does not. However, there are workarounds.

Table 9-1

G code	Mach3 LinuxCNC
G28	Go directly to the position stored in parameters 5161–5166.
G28 X~ Y~ Z~ A~ B~ C~	Go to the absolute machine coordinates of the position specified, then go directly to the position stored in parameters 5161-5166. This is the same as **G28**, but the Controlled Point moves to a specified intermediate position before going to its final position.
G28.1	Store the absolute machine coordinates of the current position in the parameters 5161–5166.

G30 Go directly to the position stored in parameters 5181–518.

G30 X~ Y~ Z~ A~ B~ C~ Go to the absolute machine coordinates of the position specified, then go directly to the position stored in parameters 5181-5186. This is the same as **G30** but the Controlled Point moves to a specified intermediate position before going to its final position.

G30.1 Store the absolute machine coordinates of the current position in the parameters 5181–5186.

What follows only applies to LinuxCNC:

- Both the **G28** and **G30** commands work in absolute machine coordinates and ignore any offsets such as those set up by touching off.
- **G28** makes the Controlled Point move to a predetermined position that can be set by a **G28.1** command.
- **G28.1** copies the current position into six system parameters. These are simply storage locations that the system uses and, for this command, they are numbered 5161 to 5166 so that they can store values of X, Y, Z, A, B and C.
- **G28** will then move the Controlled Point straight to the position stored by **G28.1**.
- **G30** makes the Controlled Point move to a predetermined position that can be set by a **G30.1** command.
- **G30.1** copies the current position into six system parameters, numbered 5181 to 5186.
- If LinuxCNC has been appropriately configured and an **M6** command is issued, whether in MDI mode or in a program, the Controlled Point will move straight to the position stored by **G30.1**.

M6 TOOL CHANGE SET-UP

When LinuxCNC starts up, it reads the values of a set of parameters in a file created when you set up the software for your own mill. One of the parameters in that file controls what happens when you use an **M6** tool change command. That file will have the name you gave it and the file extension .ini, and it will normally be stored in your Home directory, in LinuxCNC/configs.

Open that file in a text editor (like gedit) and look for the section headed [EMCIO]. The parameter TOOL_CHANGE_ AT_G30 can be set to 1 to make the Controlled Point go to the **G30** location when it meets an **M6** command. Setting that parameter to 0 makes the software ignore any **G30** moves when it meets an **M6** command.

If you make changes, save the file. To make changes take effect, restart Linux-CNC.

Here is one way of using **G28** and **G30** commands to help with tool changes:

- Move the Controlled Point to a safe position above the work. Make this a position that can be reached from any point on the workpiece without hitting an obstruction. A simple move to safe Z would work well.
- Type **G28.1** to store that position. This position is stored using absolute coordinates, so it is not complicated by work offsets.
- Move to a safe position somewhere above the table, clear of obstructions, where it would be convenient to change the tool.
- Type **G30.1** to store that position.
- Now go back to the original safe position in relation to the workpiece by typing **G28**.
- Type **T2 M6**.

When the software encounters the **M6** command, it will move the Controlled Point straight to the G30 tool change position. Change the tool and click OK in the Toolchange window.

So a useful sequence for a tool change might be:

- Move to safe Z.
- **G28.1** (store that position).
- **T3 M6** (go to the **G30** position to allow the tool to be changed).
- **G43** (to use the tool offsets for that tool).
- **G28** (to return to the **G28** position).

Tool Holders and Tool Changers

Once a tool length offset is given to a tool, the position of the Controlled Point in relation to the end of the spindle must not change, so each tool must stay in a holder of some sort until the whole job has been completed. This means setting each tool in a tool holder of its own that can be put back into a known position in relation to the end of the spindle each time the tool is to be used. This really means using multiple tool holders that locate on a taper or locate against the end of the spindle. Although there will be an extra cost, this is more than offset by the saving in time during tool changes by removing the need to touch off every time a tool is changed.

Tool lengths and the required offsets can be determined off the machine, using a jig and a height gauge or dial test indicator or some similar method, for convenience.

USING CUTTER COMPENSATION

The difficulty in programming a path for a cutter running around the outside or inside of a shape is that the path of the Controlled Point does not follow the outline of the shape.

Instead, the Controlled Point must be offset from the shape by the radius of the cutter to the outside or the inside, depending on whether the result is to be a finished profile or a pocket. This means extra calculation in producing the path for the Controlled Point. It also means that if a different size of cutter is used, the path of the Controlled Point must be recalculated. Although the plan might be to stick with the same tool, that tool might not be available. There is an additional complication in the real world when a tool cuts undersize or oversize, perhaps because it has been resharpened.

For many jobs, these problems can be overcome by always programming the Controlled Point to move along the path defined by the shape of the features on the object, but making the CNC program add or subtract an offset when cutting the outside or the inside of those shapes. This is known as a cutter radius offset and it is programmed using the **G41** or **G42** commands, depending on whether the cutter is to be offset to the right or the left of the outline of the workpiece. Fig. 9-2 shows a shape consisting of an outline (shown in blue) and a pocket (shown in orange).

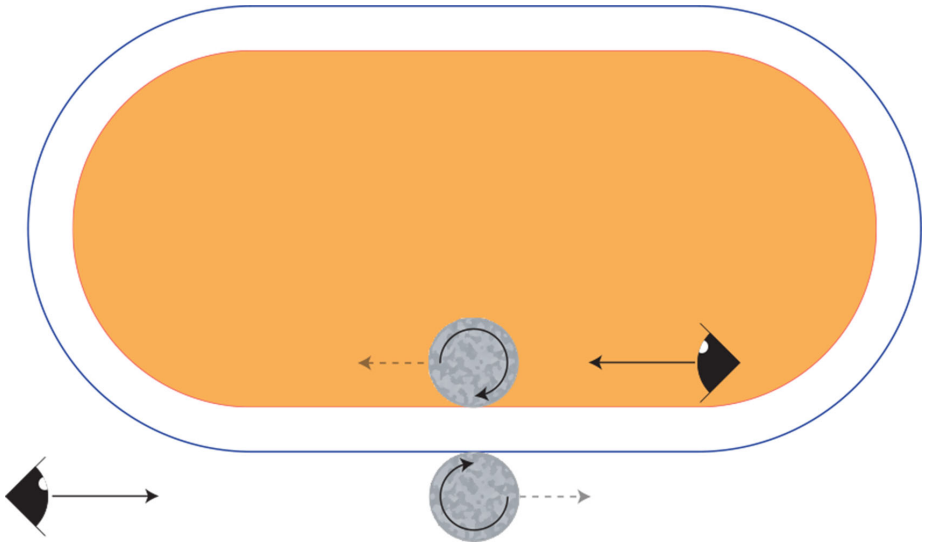


Fig. 9-2 The direction of travel of the cutter, viewed from behind, determines the appropriate command to use to compensate for the diameter of the cutter.

For the outside of the object, the path of the Controlled Point is offset, by the radius of the cutter, to the outside of the blue outline and the cutter will travel around the outside of the outline, cutting conventionally. Standing behind the cutter, watching the cutter move away along the job, the Controlled Point will be to the right of the outline for conventional milling, so the **G41** command should be used.

If the action of the cutter was to climb mill and travel clockwise around the outside of the shape, standing behind the cutter as it moves away would put the Controlled Point on the left of the path of the blue outline and **G42** should be used.

Similarly, for the pocket, if a conventional milling cut is used, the cutter will travel clockwise around the inside of the pocket, and standing behind the cutter as it moves away means the Controlled Point will be on the right of the edge of the pocket (G41). Cutting the edge of the pocket using a climb milling cut means the Controlled Point would travel around the shape in an anticlockwise direction and, seen from behind, would be on the left of the edge of the pocket (**G42**).

The form of these commands is: **G41 D<tool number> P<tool radius>**.

The D value refers to the position of a tool in a tool carousel (usually of an automated tool changer) or a predefined tool in a set of values in a tool table. D and its value can be omitted unless you intend to change the tool.

P is an optional value for the radius of the tool that will override the associated value of the radius already in the tool table.

So **G41 P5** will offset the Controlled Point 5 units to the right of the programmed path (for a cutter 10 units in diameter). **G42 P3.5** will offset the Controlled Point 3.5 units to the right of the programmed path (for a cutter 7 units in diameter).

G40 will cancel all cutter radius compensation and is a useful command to put at the start and end of every program. That way, you will always know that the Controlled Point will travel along the path described by movement commands, at least until you apply the cutter radius compensation that suits your purposes within that

program. Ending with **G40** also means that there is no possibility of unwanted cutter compensation inadvertently being applied from a previous program to the current program.

Cutter radius compensation means it is possible to program movements along the geometry of the shape without having to perform mental gymnastics, and it provides an easy solution to the maths, changes of cutter, and calculation of curved paths around sharp corners. However; there are some important points to bear in mind.

Cutter compensation must be accompanied by entry and exit moves at least as long as the diameter of the cutter. This means you must add linear or curved entry and exit moves, as appropriate. Fig. 9-3 illustrates this using lead-in curves that are long enough for the cutter to move to an offset position before cutting of the faces begins.

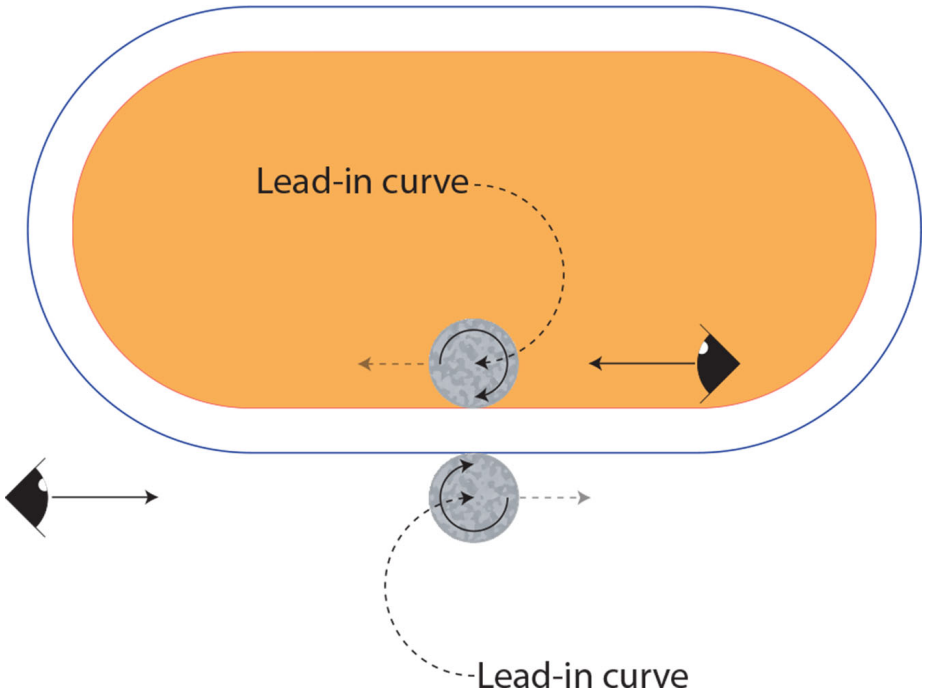


Fig. 9-3 Lead-in curves help smooth the entry of the cutter into the work and help deal with the effects of backlash.

Cutter compensation affects the following commands:

- **GO**and **G1** (linear movement);
- **G2** and **G3** (movement along an arc);
- **G53** (movement in absolute machine coordinates);
- **G54** to **G59.3** (coordinate systems);
- **G81** to **G89** (canned cycles).

This means that using and cancelling cutter radius compensation does require some care in planning the whole of the program.

It is worth noting that one important aspect of CAM programs is that they take care of cutter movement and cutter radius compensation, usually by calculating the path of the Controlled Point required to produce the shape being machined without using the **G41** command. This means that there is no further need to deal with cutter compensation. Changing the diameter of the cutter simply means selecting an appropriate tool from the library and making the CAM package recalculate the toolpath.

Many of the parts on a clock can be enhanced by engraving.

10 Engraving

In this chapter, you will learn:

- how to use engraving cutters to engrave flat surfaces;
- how to use a diamond drag tool to engrave work;
- how to produce an engraving from a photograph.

Machining a part to size and producing a good clean finish is usually all that is required for functional purposes, but embellishing the part by engraving a design or some text adds an artistic dimension that can really bring the part to life.

Sometimes, engraving is used to create labels and legends, but it is more often used to add a logo or a pleasing design. And somewhere in between are the clock and watch dials that carry numerals for functional purposes but add an artistry to their presentation that transforms their look and seems to add personality to the whole object.

Engraving is an old traditional art that depends on skill and lots of practice, but it can be carried out with relative ease on a CNC mill or purpose-made engraving machine. As with many traditional craft skills, engraving has long been practised with the assistance of devices like the pantograph engraving machine, but it is a skill that has been changed even further by the use of CNC machinery. Engraving using a CNC machine means the challenge is no longer in the manipulation of a tool using the hand and the eye or in following a template, but in the programming of the machine.

Engraving takes place using the same CAD/CAM/CNC cycle as for any other workpiece, but the CAM stage is almost always carried

out with the aid of a CAM program, rather than by hand coding using G codes. The keen programmer might be able to work out the toolpaths for a simple engraving pattern composed of straight lines and simple circular curves, but for anything like a more complex traditional engraving pattern or for anything containing text, a CAM program is essential.

The rest of this chapter assumes you will use a program like Cut2D, VCarve Pro or Aspire, or any of the many similar programs that cater for engraving of text or patterns. All are suitable for creating engravings. VCarve Pro and Aspire are capable of 2½D and low-relief 3D carving, while Cut2D is restricted to flat 2D surfaces and 2½D pockets and profiles, although that does still allow text and complex shapes to be engraved, the limitation being that Z axis movements are not simultaneous with X or Y axis movements so the tool plunges but does not penetrate or exit the work at an angle. The consequence is that 3D bevelled edges cannot be machined, and exit corners (for example at internal corners of letters) cannot be formed as sharply using Cut2D as with VCarve or Aspire. Of lesser importance is the fact that the design tools within Cut- 2D are less extensive than in VCarve Pro or Aspire.

There is more than one way to engrave and the two principal methods are rather different, but we will begin with the form of engraving that uses a cutting tool as that is capable of the most sophisticated results.

ENGRAVING CUTTERS

Engraving tools for the milling machine are tapered tools that have cutting edges on one side and on the bottom of the tool (see Figs 10-1 and 10-2). The angle of taper and the width of the point are chosen to produce the effect required from the tool, and the tool cuts a channel with angled sides as it moves in X and Y. Engraving cutters are normally specified by the half-angle of the sides and the width of the tip, although if you are buying from abroad, via one of the popular internet auction sites, you would be wise to check the

meaning of the angle quoted by the supplier as some quote the total included angle rather than the half-angle.

RELIEF CARVING

A carving created *in relief* provides an illusion of three-dimensionality by machining away the background so that the subject of the carving stands out from the background, although it is not raised up further than the original surface of the workpiece. In *high-relief* carving, the background is machined away to a considerable depth, while in *low relief* the same technique is used but the background is not machined away to such an extent. An architectural frieze around a building may be in high relief, while the sculpting on a coin is usually in low relief.

Fig. 10-1 A selection of engraving cutters.

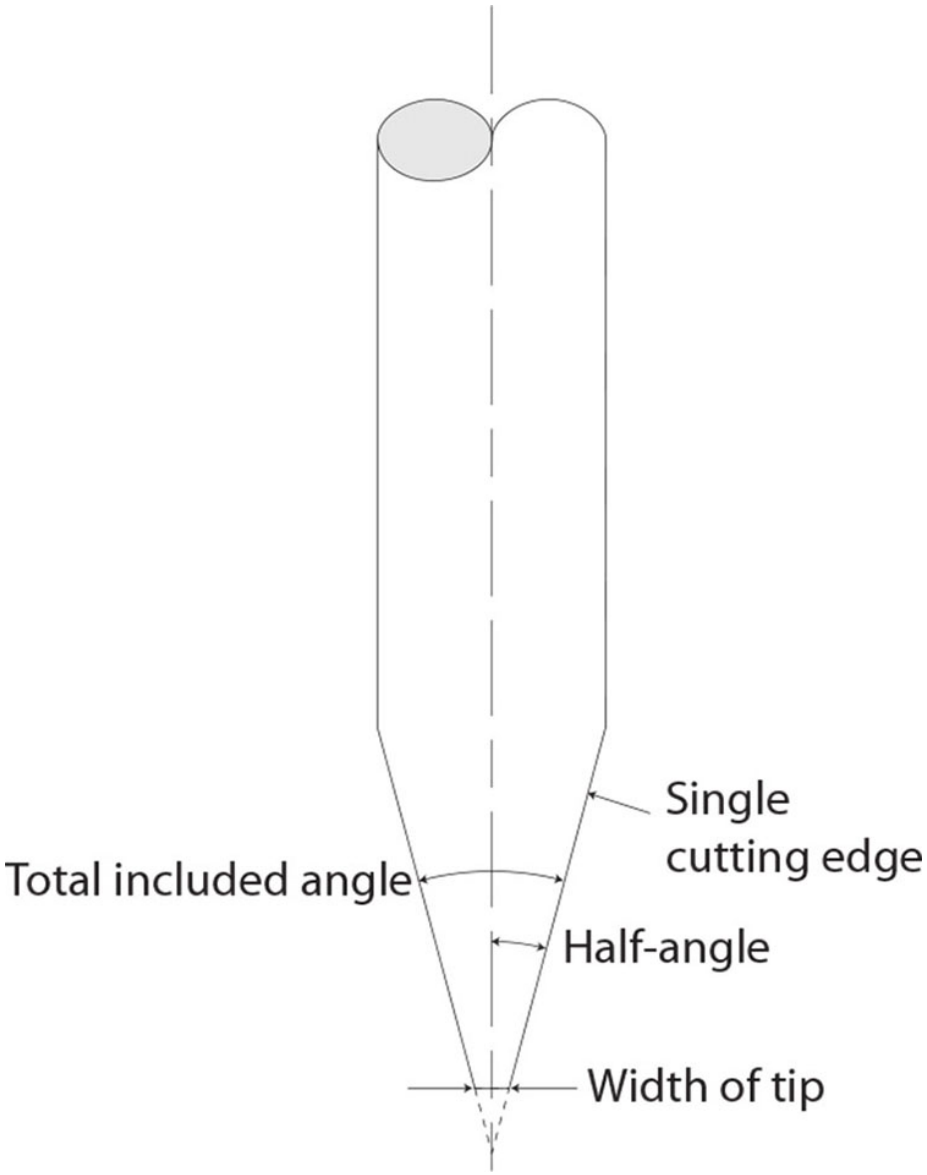


Fig. 10-2 The important features of an engraving cutter.

The depth of cut determines the width of the channel, which can vary from very fine to quite coarse so that a range of visual effects is possible, as well as some machining operations that have nothing to do with traditional engraving.

Fig. 10-3 shows a simple application of engraving in which a legend is added above a set of sockets. Fig. 10-4 shows a more complex decorative pattern; while Fig. 10-5 shows text engraved in a circular path.

Fig. 10-3 Engraving on a die-cast box

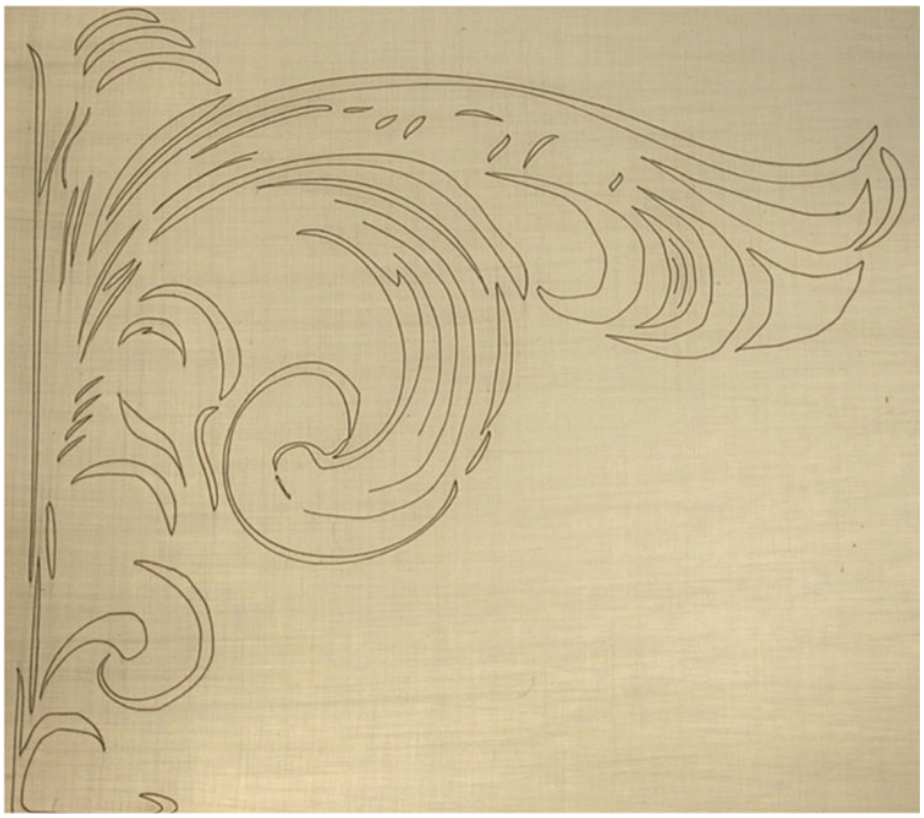


Fig. 10-4 A decorative pattern produced by engraving.

Fig. 10-5 Text engraved on a circular path around a clock wheel.

A basic requirement for engraving using a conventional cutter is that the workpiece being engraved must be flat. Any variation in flatness will be apparent as a variation in the thickness of the lines or edges in the engraved pattern. Although this might be very small, the eye is a remarkably sensitive mechanism and will inevitably pick this up. Fig. 10-6 shows a die-cast box measuring some 250 × 250mm in which the manufacturing process has produced a very small but complex dip towards the corners of the lid (note the light visible against the bottom edge of the ruler) that will have a visible effect on the depth of engraving cut. Fig. 10-7 shows the use of some large washers to selectively pull parts of the lid more tightly on to the backing material to help compensate and produce a tolerably flat surface. This is very much a workaround and is best avoided. Although it might make more sense to pull the back of the lid into close contact with a flat surface, the underside of the lid has die ejection marks and embossed lettering and symbols so that it is not flat and may not be completely stable against another surface.

Fig. 10-6 The dip in the lid of a large die-cast box.

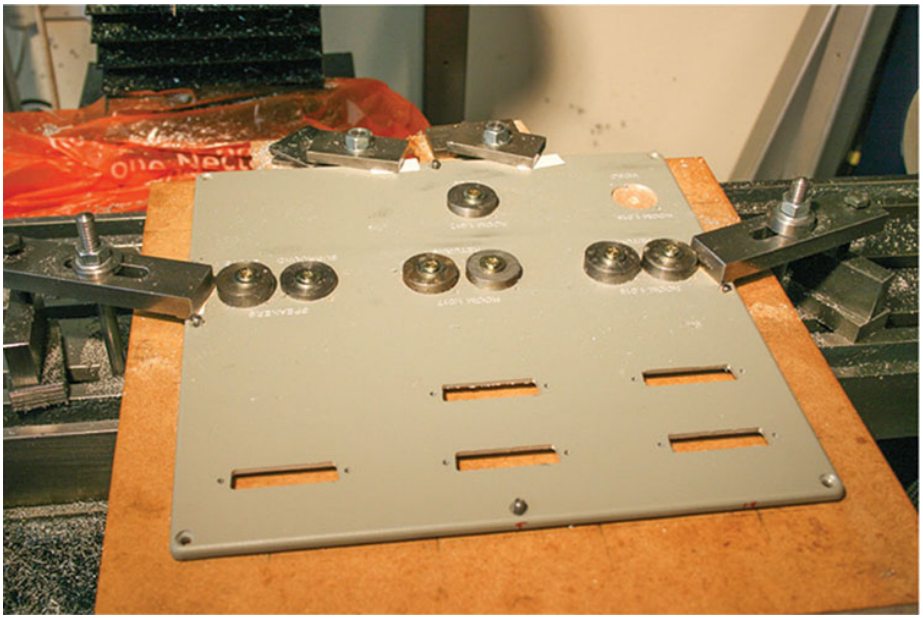


Fig. 10-7 Die-cast box lid pulled straight using large washers as clamps.

One way of ensuring thin work is held flat is to use a vacuum table (Fig. 10-8) where the work is sucked against the table, partly to hold it in position, but mainly to ensure it is held uniformly flat. Although this is a relatively expensive solution requiring a table and

a vacuum pump, it can provide a sufficiently powerful pull of the work against the table that other clamps are not necessary, simplifying the set-up of the work.

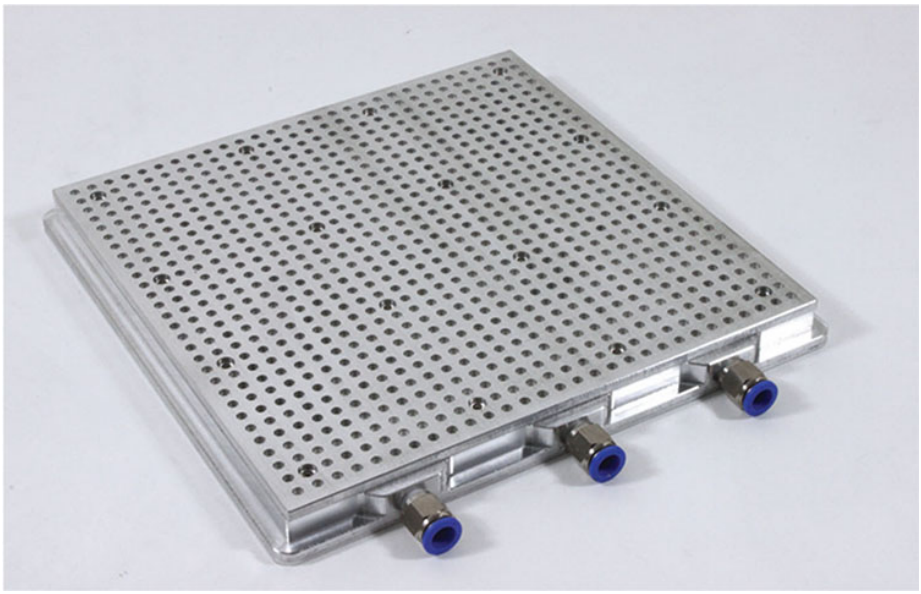


Fig. 10-8 A vacuum table. Photo courtesy of Stritzelberger Steuerungstechnik GmbH.

The work is placed on the vacuum table and the uncovered holes are covered with a rubber mat to seal the holes and/or the area under the workpiece is sealed with rubber strips placed in recessed channels in the table. It all depends on the model of table.

ENGRAVING LINES

Straight lines are easily engraved, although the depth of the lines determines the approach. Shallow lines may be engraved with a single pass, but given the fragility of narrow engraving cutters with small included angles, narrow lines of any appreciable depth should be produced using multiple passes, and at each pass the tool should be ramped carefully into the work.

Bearing those factors in mind, engraving cutters can be treated as shaped milling cutters and toolpaths programmed in just the same

way to produce lines and arcs as required. However, it is just as easy to use software to assist with the programming, even when the cutter paths are simple straight lines.

Lines and patterns can be drawn in a CAD package, but most CAM packages allow drawings like this to be created within the package, which speeds up the process. Simply draw the lines in a CAM package, specify the engraving cutter to be used and the depth that will give the required width of line, then machine the work.

Toolpath previews vary in accuracy from one software package to another, but a good preview facility will give an accurate indication of the final look of the work, so it is often enough to experiment with the depth of cut and the taper and width of the cutter to obtain a pleasing result on screen before machining. For more accuracy, the depth of cut can be calculated and, provided the Z origin and initial tool tip Z position are accurately set, a predictable result should appear after machining.

To calculate the width of the cut line, use: line width = tip width + 2 × depth × Tan A degrees, where A is the half-angle of the cutter in degrees.

To calculate the depth of cut required for a given width of line, use: depth = (line width – tip width)/(2 × Tan A degrees).

As a guide, a depth of cut of 0.1 to 0.2mm (0.004 to 0.008in) is a good place to start with cutters with half-angles of 15 degrees or more, unless you know your specific requirements.

To produce graduations on scales, take care to specify the positions of lines accurately, using coordinates. Placing lines by eye on a computer monitor is seldom sufficiently accurate for a scale. Unless your CAD/ CAM program allows lines to be positioned accurately using coordinates, this would be a good application of a little simple G code programming.

Wrapped scales can be created by calculating the length of material needed to wrap completely around an object like a cylinder, then engraving this as a flat object before attaching it to the cylinder. This is not as straightforward as it seems because of the thickness of the material used for the scale; a better technique is to use a fourth (rotary) axis.

Using flat material, add half the thickness of the material to the radius of the cylinder around which the scale will be fixed, making an additional allowance for all but the thinnest glue. Then use the formula: length = 2 × pi × radius. Pi is an awkward number and has an infinite number of digits after the decimal point. To three places of decimals, its value is 3.142, but that is an approximation, although it is good enough for thin glue.

Why use half the thickness? Fig. 10-9 shows a side view through a flat plate. When the plate is bent, it bends around its **neutral axis**, which is normally in the centre of the section. The underside will be bent less than the neutral axis and is actually in compression, being squeezed to occupy less space that the material at the neutral axis. The top side must stretch more than the neutral axis. In theory, the ends will meet neatly when the plate is wrapped around a cylinder. In practice, the plate may not be sufficiently flexible to allow the ends to meet perfectly or to lie completely against the cylinder, so it is wise to experiment a little with the length and to be prepared to chamfer the bottom edges of the plate (Fig. 10-10).

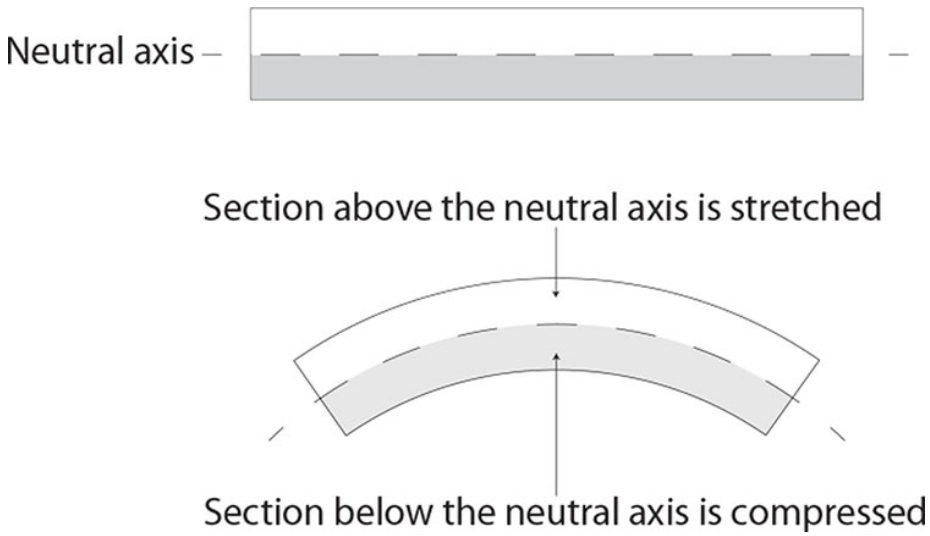


Fig. 10-9 The neutral axis of a thin plate.

Engraving scales around a cylindrical knob (Fig. 10-11) can be done directly using a fourth (rotary) axis and a short piece of G code.

Not all CAM packages are capable of generating code to control a rotary axis and for those that do, it is important that you understand what the package is trying to do with that axis and where that package expects the rotary axis to be in relation to the Controlled Point.

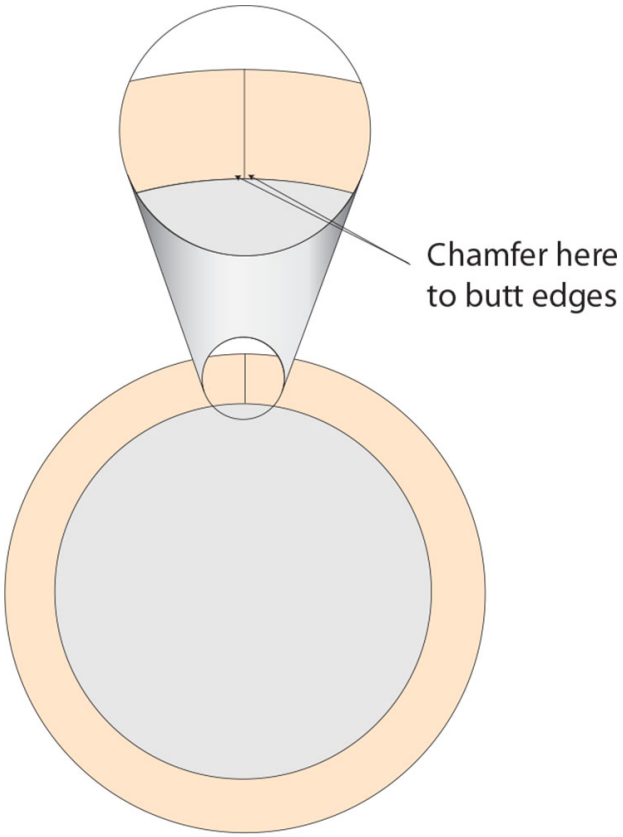


Fig. 10-10 Edges that may need to be chamfered when an engraved scale is wrapped around a cylinder.

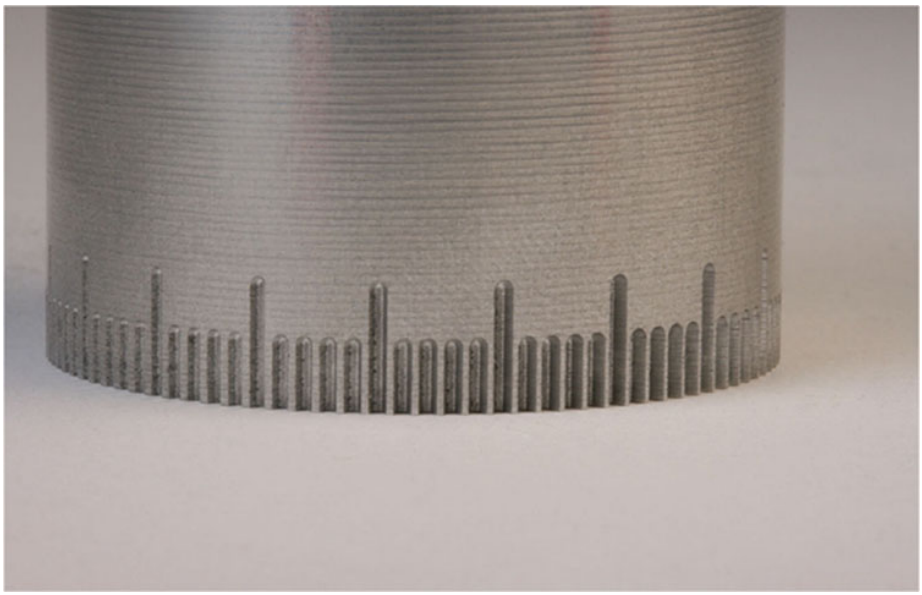


Fig. 10-11 A scale comprising lines engraved around a cylinder.

In VCarve Pro, for example, a flat drawing can be wrapped around an axis using an additional piece of software called a **gadget** that works with the main program but extends its capability. A gadget is rather similar to a wizard, but is an external program that is used to do a specific task and works in conjunction with the main program. The Wrapped Rotary Axis gadget allows a scale, or any other design drawn on the flat, to be engraved around a cylinder held in a fourth axis.

GADGETS

In Vectric software, a gadget is rather like a wizard in Mach3. Gadgets are additional programs that perform specific tasks. The Wrapped Axis gadget, for example, provides a flat work area on which a design can be created and then uses this to create a toolpath that will machine that design wrapped around a cylinder. The support section of www.vectric.com provides some good examples of how gadgets work.

ENGRAVING TEXT

Engraving text is, in theory, quite simple. Just position some letters on a design, then generate the code to make a cutter follow the path using the code, then carry out the machining. In a package like Cut2D, this is exactly what happens.

At the design stage, letters can be chosen from any of the computer's TrueType or OpenType fonts as well as a range of **single line** fonts (single line fonts are often called **stick fonts** or **engravers' fonts**). The definition for an individual character is called a 'glyph'. TrueType fonts are commonly found on Macs and on PCs running Windows; glyphs contain the mathematical definitions of the lines and curves in the finished character.

OpenType is a development of TrueType and works in much the same way, holding character definitions (glyphs) as lines and curves. OpenType fonts contain a little more information about some typographic features than TrueType fonts, but both are just as useful for CNC machining.

In Linux systems, FreeType emulates True Type and OpenType glyphs, and OpenType in particular is designed to be cross-platform on Macs, Windows and Unix/Linux systems.

Because CNC systems are good at handling lines and curves, and more than capable of handling mathematical information in glyphs, machining characters is quite easy. In fact, what looks much more complex than most typical CNC-machined workpieces is an illusion, and type is no more complex to machine than any other work.

Single line fonts have characters that can be produced using single lines. The thickness of the lines determines the look and the size of the font.

Fig. 10-12 shows characters drawn using a TrueType font and the same characters drawn using a single line font. The single line character can be engraved quickly and is perfectly legible, but the TrueType character has an outline and a thickness, and can be

given a texture within its outline that can really bring the character to life. It all depends on the effect that is required for the job.

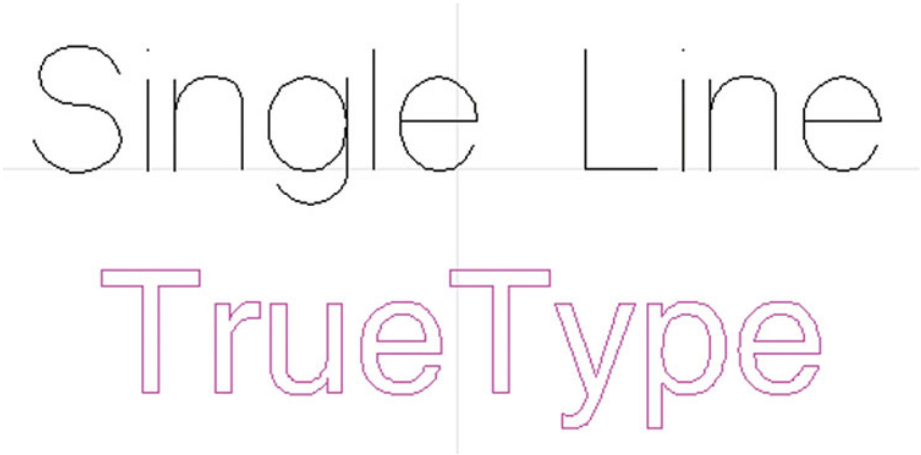


Fig. 10-12 TrueType characters and their single line equivalents.

That does not mean TrueType fonts are always better than single line fonts, because the single line fonts lend themselves particularly well to **diamond drag** engraving (see later). Single line font characters can also be given the illusion of thickness by using a deeper cut.

Whether single line or TrueType, fonts are really just shapes and can be machined in the same way as any other shapes. What is different about fonts is that the glyphs (characters' definitions) can be relatively complex, especially for fonts that are part of a related family, like Times Roman, Helvetica or any of the many common typeface families. The glyphs have already been defined, though, and the corresponding shapes can be reproduced by using software to read the glyph definitions. The power of good software is in the way it can read the files, reproduce the definitions of the shapes, position the characters accurately and allow manipulation of the character shapes. So characters can be magnified or shrunk, and stretched, reflected or otherwise distorted. Fig. 10-13 shows an example of part of a design in which the characters have been stretched to create a unique shape.

Fig. 10-13 A sign based on stretched text.

The corners of text present a particular challenge because, although the letter may require a particular depth of cut for appearance, this sets a minimum radius for the corners of the letter that may be too large to reproduce sharp corners, as shown in Fig. 10-14. Good software can compensate for this by varying the depth of cut in the corners, effectively ramping the cutter out of the work as it moves towards the corners. Reducing the depth of cut reduces the minimum radius because of the taper on the engraving bit so the minimum radius becomes not much more than the width of the tool tip at those points, giving a much crisper look to the character. This would be complex to program by hand, but is effortlessly accomplished within good CAM software.

Fig. 10-14 The radius of a cutter may prevent some detail from being reproduced. The blue circles indicate the closest the cutter can approach a sharp corner.

Reversed-out Lettering

Reversed-out lettering is produced by milling away the material outside the letter outlines to leave the letters standing proud, as in a locomotive nameplate or a house sign, as shown in Fig. 10-15. Depending on the material and the cutting conditions, the machined portion may show tool marks, as in the case of the clock key shown in Fig. 10-16. With wood or other material, the bottom of the cut section can be sanded to finish the work. With metal or with awkwardly small work, finishing the bottom of the work is not so easy, but filling the engraved recess gives an attractive effect as shown in Fig. 10-17. Fillers include resin, wax and paint.

Fig. 10-15 A sign incorporating reversed-out lettering.

Fig. 10-16 Tool marks inside the pocketed areas of a clock key.

Fig. 10-17 Filling pockets with paint or another filler hides the tool marks and enhances the visual effect of an engraving.

Bevelled Lettering

Although reversed-out lettering is visually attractive, the sides of the raised letters are almost straight, being sloped simply by the angle of the cutting edge of the tool.

Bevelled lettering has a much more pronounced angle on the sides, often leading up to a ridge along the centre of the shape of the letter (Fig. 10-18). This effect is produced by using a tool with a half-angle matching the bevel required on the letter. Cutters with 45-

degree half-angles are commonly used in this application, but the exact angle depends on the effect to be produced.

IFig. 10-18 Large bevelled lettering for a large sign. Photo courtesy of Enseignes Bois et Passions.

Text on a Cylinder

Engraving straight lines on a scale around a cylinder is straightforward using either G code or a suitable CAM package, but including letters or numerals on a wrapped scale is best done using a CAM program.

In VCarve Pro, the Wrapped Rotary Toolpaths gadget not only wraps lines, but can wrap any accompanying text or any design around any cylindrical object. Simply add the text to the scale as it is drawn on the flat and then let the software wrap the whole drawing around the cylinder.

Fig. 10-19 shows an example of numerals added to a wrapped scale.

There are some limitations on what Wrapped Rotary Toolpaths can be used for, but these do not affect engraving on to a cylinder.

Fig. 10-19 An engraved scale comprising lines and numerals wrapped around a cylinder.

KNURLING

Knurling can be carried out using a set of knurling wheels, but that is not always effective in soft metals like aluminium or brass. In those metals, knurling can be simulated very effectively using an engraving cutter to cut V-grooves around the circumference of a cylindrical object.

Decide first on the number of grooves. This can be done by calculating an approximate number of grooves or divisions around the circumference, then adjusting the depth of cut to give a convenient number. Or decide on the number of grooves and then adjust the depth of cut to give a pleasing appearance.

Basic straight lines are easy to do (Fig. 10-20), and using an end mill instead of an engraving cutter can give a pleasing result, as shown in Fig. 10-21.

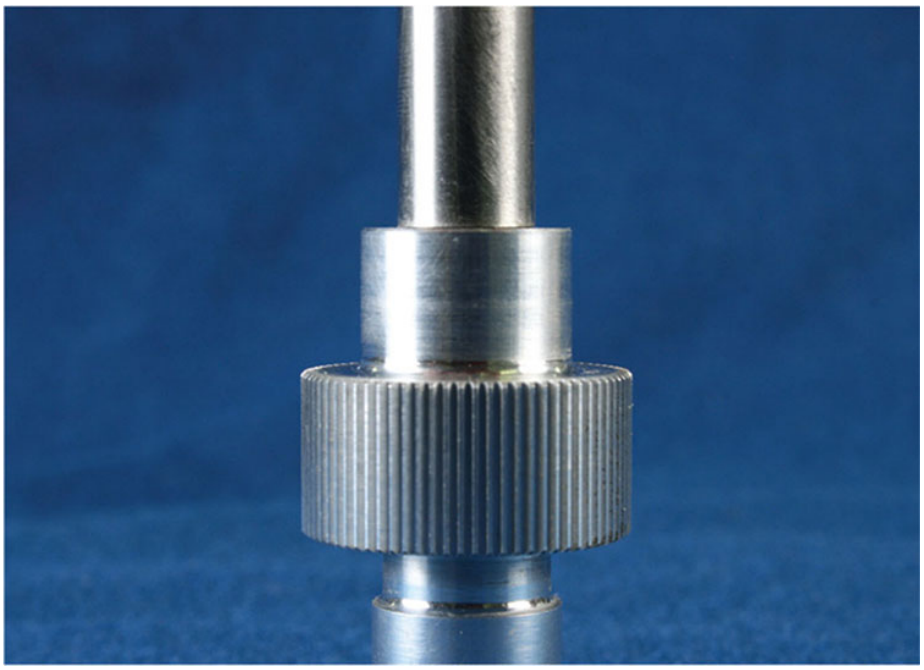


Fig. 10-20 Straight knurling can be created by engraving straight lines.

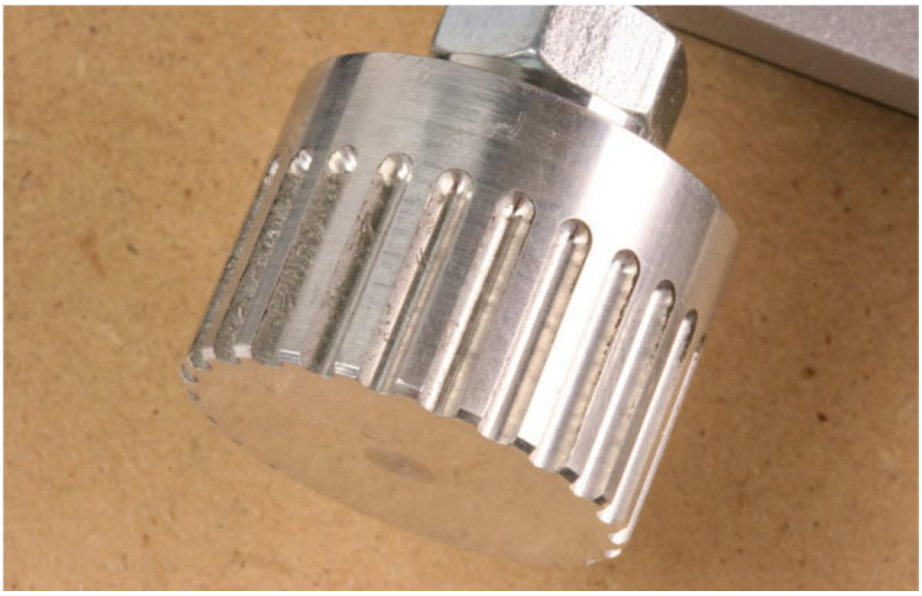


Fig. 10-21 Flutes can be created using a ball-nosed end mill.

The same job could be done using CAM software such as VCarve. Once the number of grooves is known, they can be

produced by first using the Wrapped Job Setup gadget to set up the job to be machined on a rotary axis, then using the Wrapped Fluting Layout gadget to allow the grooved paths to be defined precisely. The toolpaths can then be calculated and machining can take place using the rotary axis.

Rotating the fourth axis during an otherwise straight engraving cut simulates diamond knurling rather well, and the depth of cut can be controlled to give a good grip as well as an attractive finish (Fig. 10-22).

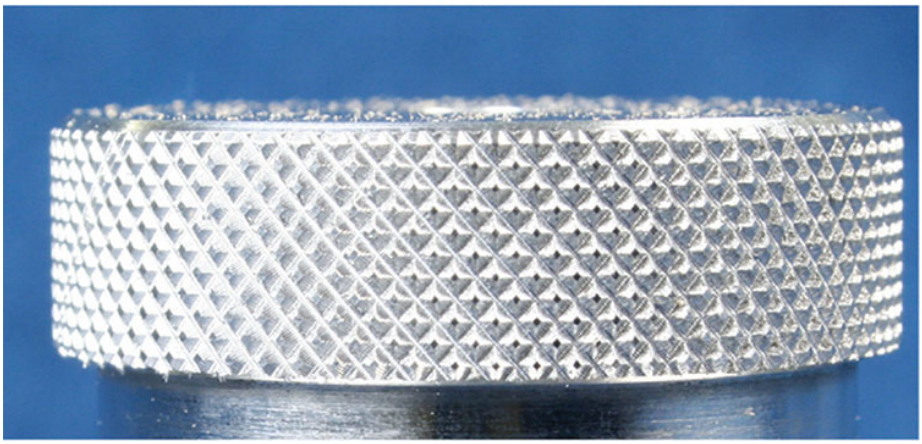


Fig. 10-22 Diamond knurling can be created around a cylinder by rotating the cylinder as the engraving cutter moves in a straight line.

In VCarve Pro, the Wrapped Spiral Layout gadget can be used to define a set of spiral paths and then a pattern can be reproduced along that spiral path. This allows more complex patterns to be engraved on to a cylinder so that, instead of simple grooves, more complex movements of the axes are used to produce flowing designs around the cylinder; and instead of the cylinder having a uniform cross section, it may be tapered from one end to the other. The applications are limited only by imagination and the features of the CAM software being used.

DIAMOND DRAG ENGRAVING

The term diamond drag engraving suggests what is involved: a diamond-tipped tool is dragged across the surface of the work to scratch a design into the surface. The spindle remains stationary and the tool does not revolve. Fig. 10-23 shows a typical diamond drag tool with a tiny diamond embedded into the tip of the tool. Because diamond is a hard substance, diamond drag engraving can be used on a wide range of materials, giving its crispest results with medium to hard surfaces.

Fig. 10-23 Diamond drag engraving tool.

A diamond drag tool contains a spring that is preloaded, usually by turning the body of the holder or adjusting a screw. The Z axis is then set to zero with the tip of the tool against the surface of the work and a cut is applied along the Z axis. The tool is then dragged across the work, using exactly the same technique as for a cutting tool, and the pattern appears as the diamond scratches the work. Adding a typical ‘cut’ compresses the spring, adding pressure on the tool tip against the work.

As with conventional engraving cutters, diamond drag tool points have a tip width and a taper angle, although there is a much smaller range of widths and angles than for conventional cutters. Useful sizes tend to be 90-degree and 120-degree tips (that is the total included angle; or 45-degree and 60-degree half-angles), with the 90-degree point being the most useful for general work. The 120-degree point gives a wider scratch for the same depth.

Typical depth of cut might be 0.2 to 0.5mm, with a speed of movement from 250mm/minute to 1,000mm/min. The depth of cut varies considerably, depending on the initial set-up of the spring inside the tool and the hardness of the material, and some manufacturers suggest ‘cuts’ of 2mm or more.

The tool drags rather than cuts, so the depth of cut is small compared to what is possible with an engraving cutter, and the engraved lines have a small burr on either side. This has the interesting effect of enhancing the visibility of the engraving as the burrs catch the light. Rubbing the work with an abrasive to remove the burrs will emphasize the shallow depth of the lines, so polishing needs to be done with care.

One advantage of diamond drag engraving is that because the tool holder contains a spring, the diamond tip will tend to follow gentle undulations in the surface of the work. This also allows shaped work to be engraved. The Z movement does need to track the undulations as far as possible, but the spring allows the tool to cope with small variations. This ability to track varying height is also useful when engraving rotating work, such as a cylinder or a goblet shape.

Another advantage is that diamond tools are hard enough to engrave glass and can mark steel and stainless steel, although the depth of cut is shallow in these harder materials.

Although a diamond tool is not suitable for clearing areas to create recesses, it can be used to cross-hatch an area or cross-hatch within the outline of a letter (Fig. 10-24), producing an artistic effect that helps bring the letter to life.

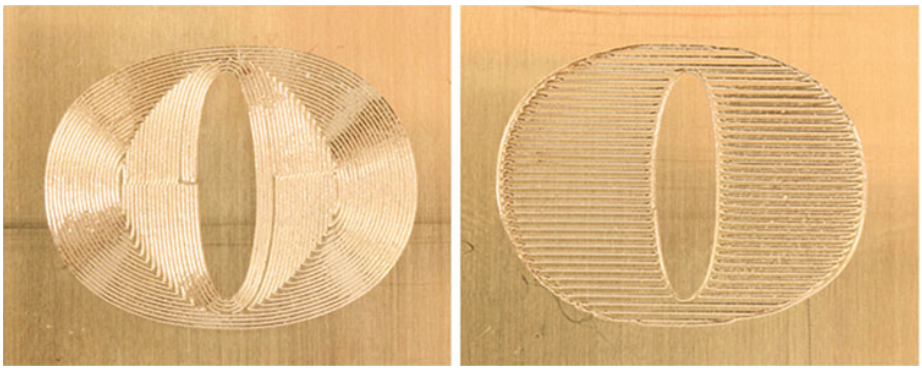


Fig. 10-24 A flat area can be filled with cross hatching using straight lines, or lines that follow the outline of the area.

ACQUIRING ART

One of the challenges for an engraver is acquiring artwork suitable for engraving. You may be lucky enough to be good at drawing and can create your own artwork, but for the rest of us, while the CNC technique might be straightforward, drawing is more difficult.

All images drawn by someone else belong to that person and are subject to copyright, so they cannot be used except by permission. Logos for company products are always copyright, as are images taken from paintings or drawings in books or galleries. However, there are some good sources of art suitable for engraving and these include art provided under a Creative Commons licence; libraries of copyright-free art (not strictly copyright-free, but placed in the public domain for anyone to use) and internet sites that provide art specifically so that you can use it for engraving.

You can also create images using a webcam or other camera attached to the computer and a drawing program; a digital camera and artwork programs; or a scanner before employing software to trace the outline of scanned material.

The Creative Commons licence system enables content such as pictures and drawings to be freely reused subject to certain conditions, which vary from one item to another, but often include an attribution (credit) to the original creator or a prohibition on

modification, or allow modification provided the result is available under a similar Creative Commons licence. Full details are available from http://creativecommons.org/.

Libraries of ‘copyright-free’ art and sites specializing in images for engraving may provide sources of files that can easily be converted to CNC use. For example, www.profitablehobbies.com provides illustrations for engravers, together with supplies such as objects to engrave.

Flickr and Photobucket contain many images of interest to engravers, and a quick search on Flickr will produce thousands of items available under a Creative Commons licence.

Libraries of images for purchase include sites like Fotosearch, 123RF and others. These sites typically offer ‘royalty-free’ art, which you buy for a one-off fee and can use thereafter. Check out the licence conditions on each site. One benefit of these sites is that they provide files for vector illustrations.

A webcam can be used to photograph objects or designs that can then be used as the basis for tracing or redrawing in the computer to produce a pattern suitable for engraving, as shown in Figs 10-25 and 10-26. A good camera that can download images to the computer can do the same job but at a higher resolution, or can be used for an object that cannot be captured using a webcam. This may allow greater control of lighting, focus and framing.

Fig. 10-25 Webcam photo of an engraved plate.

Fig. 10-26 Drawing of an engraving pattern photographed using a webcam.

A camera can also be used to photograph objects (Fig. 10-27) that can then be transferred to a program for photo manipulation, perhaps to lift the image from its background (Fig. 10-28). That image can then be manipulated and/or drawn over to provide an image composed of lines suitable for engraving (Figs 10-29 and 10-30).

Fig. 10-27 Flower photographed with a digital camera.

Fig. 10-28 Flower head lifted from the background of a photograph.

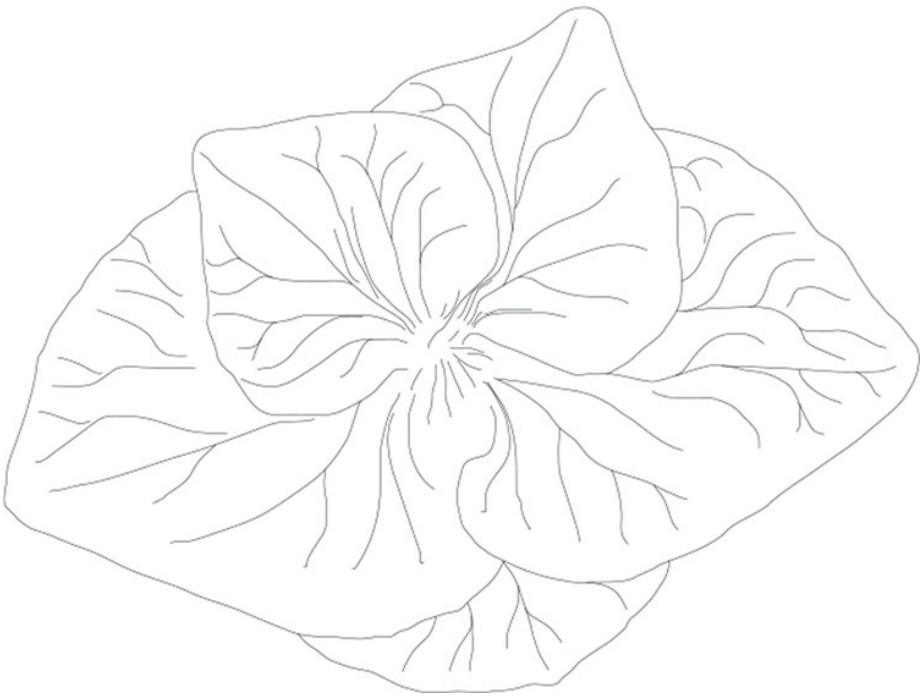


Fig. 10-29 Drawing of an engraving pattern for a flower, prepared from a photograph using a photo-editing program.

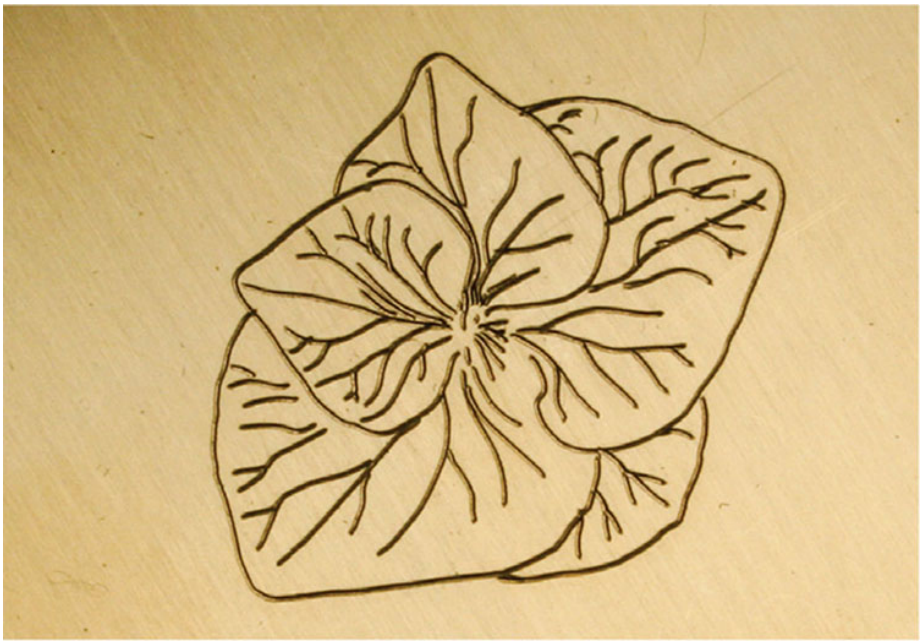


Fig. 10-30 Engraving of a flower head based on a photograph.

A scanner can also be used to transfer images to a computer so that they can be manipulated and turned into drawings suitable for engraving. Brass rubbings, for example, can provide inspiration for many a fine design, where copyright permits (Fig. 10-31).

Fig. 10-31 Engraving patterns can be captured using brass-rubbing techniques, subject to copyright permission being obtained.

ILLUSIONS OF DEPTH

PhotoVCarve is software that takes a photograph (Fig. 10-32) and turns it into an engraving (Fig. 10-33). It does this by taking a monochrome version of the photograph and calculating engraving toolpaths that vary in depth according to the shades of grey at each point in the photograph to give the illusion of light (shallower cuts) and depth (deeper cuts). This can result in visually rich images on

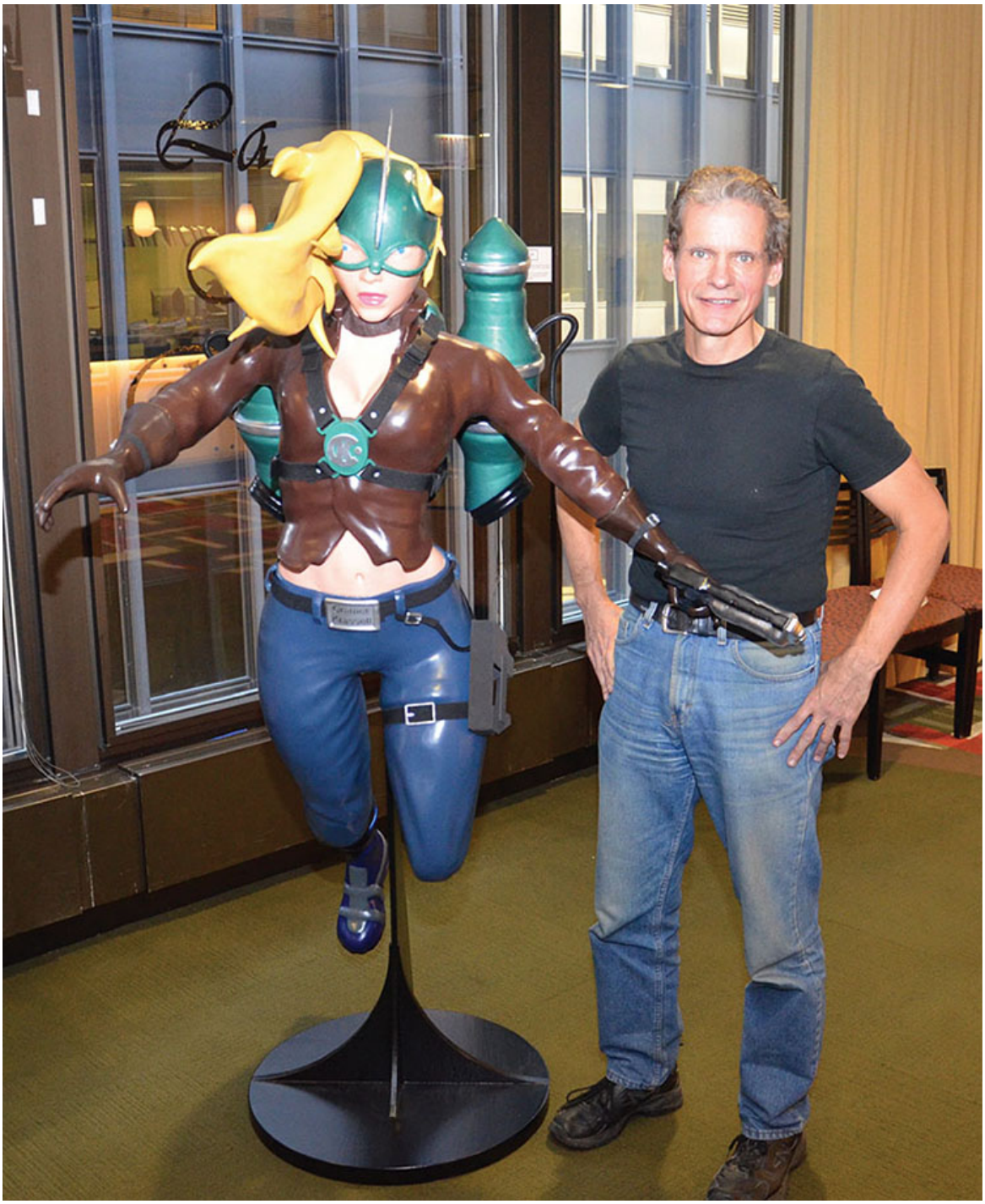
flat material. The illusion can be enhanced by staining the engraved paths to improve the contrast between the paths and the background material. This helps visibility of the toolpaths and makes the picture stand out. If the images are engraved on to thin translucent material and a strong light is placed behind the image, the effect can be very pleasing. Images can be fed into PhotoVCarve using any of the techniques mentioned above (photographs from a camera or phone, photographs from a USB camera, or images from a scanner).

A related technique is to use a greyscale depth map. A depth map uses shades of grey to represent the height of points on an object, with black areas usually at a lower height and light areas at a greater height, from a reference plane. Greyscale depth maps can be used to machine 3D objects from their maps using appropriate software. LinuxCNC has support for greyscale maps and has an image-to-gcode process that allows greyscale maps to be turned into G code to machine the object. PhotoVCarve can also interpret greyscale maps and machine the corresponding object.

Fig. 10-32 A photograph used as the basis of a photoengraving. Photo courtesy of Markus Dahlberg at Pictures In Wood.

Fig. 10-33 Photoengraving produced from a photograph by using PhotoVCarve software. Photo courtesy of Markus Dahlberg at Pictures In Wood.

Depth maps can be created by software rendering packages or from photographs and can then be applied to objects by using Adobe Photoshop or the freeware GIMP graphics program. The details of the technique vary between programs.



La Femme Rocketeer, created by Greg St. George.

11 From 2½D to 3D

In this chapter, you will learn:

- more about the differences between 2½D and 3D work.

All objects in the real world are three-dimensional; they have length, width and height (Fig. 11-1), and a CNC milling machine is likely to have three axes oriented as X, Y and Z that correspond to length, width and height.

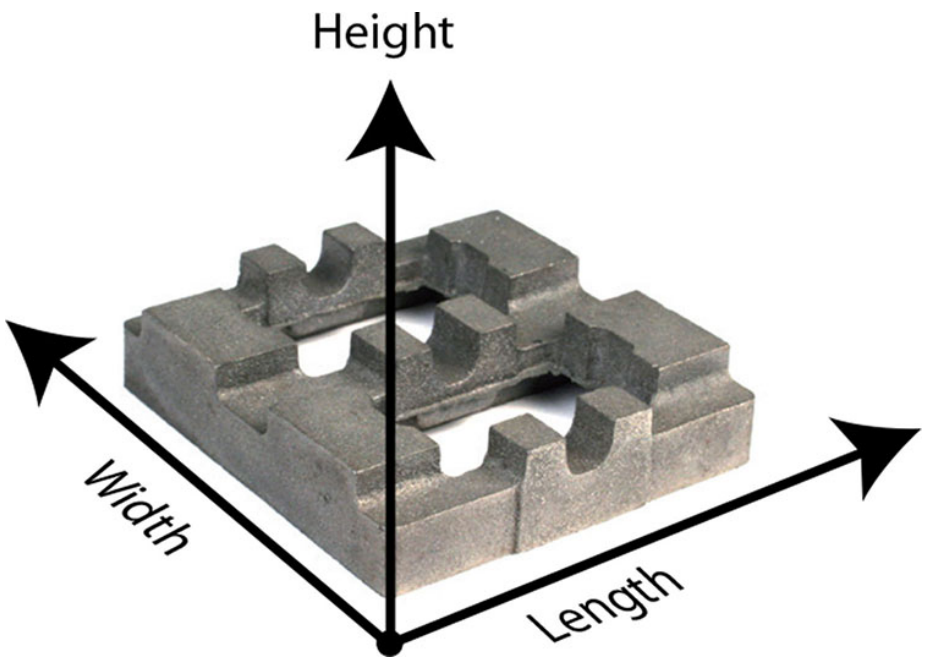


Fig. 11-1 3D objects have length, width and height.

Typically, 2½D objects feature pockets or profiles that have vertical faces, and machining a pocket or a profile in 2½D involves

vertical movements of the Z axis that are not coordinated with simultaneous movements of the X and/or Y axes.

Machining in three dimensions may require simultaneous coordinated movements of the Controlled Point in three axes, and is not restricted to vertical movements of the Z axis. There are some practical limitations to 3D machining on the typical three-axis mill, though. In the real world, a truly three-dimensional object has an ‘underside’ that may have a shape other than a flat surface, and if that 3D shape has to be reproduced on a mill, it is difficult to machine features under overhangs unless the workpiece is turned over.

Machining the object in Figs 11-2 and 11-3 from a solid block of material would involve machining downwards from the top to recreate most of the features, although some operations would require the use of a long and relatively slender end mill. Most of the features underneath the object could be omitted in this case, but the difficulty would be in reproducing the underside of the shaped portion immediately behind the flange (indicated in Fig. 11-2).

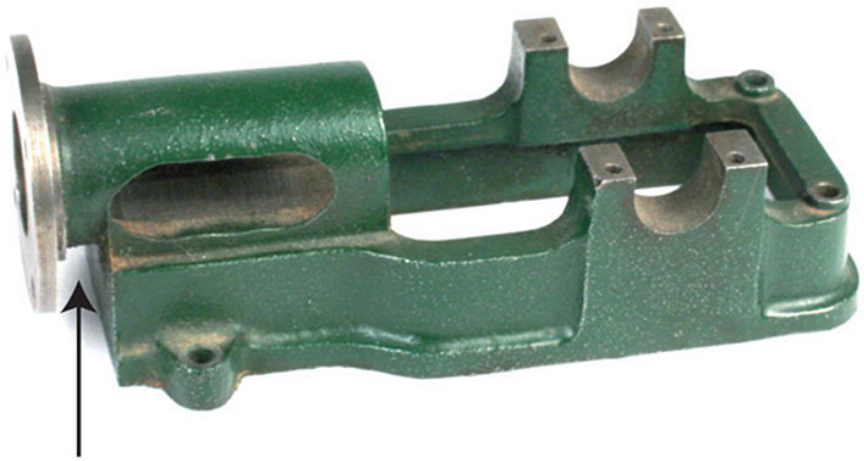


Fig. 11-2 Steam engine bedplate.

Fig. 11-3 Underside of a steam engine bedplate, showing recess detail and areas that cannot be machined from above if the bedplate is sitting on its base.

CREATING 3D MODELS

CAD software, such as Autodesk Inventor, Autodesk 123D, RhinoCAD, TurboCAD Pro and SolidWorks, exists specifically to create 3D models. Artistic modelling packages such as Blender, Silo 3D or Autodesk MAYA are also capable of creating models in three dimensions and are well suited to the creation of free-flowing surfaces, although less convenient to use by inputting precise dimensions. It's horses for courses, really, and the software tools can be chosen to suit the characteristics of the models and the inclination of the designer.

Conventional technical drawings represent three-dimensional objects in two dimensions on flat paper, but a 3D modelling program creates the 3D object within a computer system. The output can be on paper, in the form of conventional flat drawings, but the real purpose of 3D modelling is to allow the designer to think in three dimensions and to create the object directly in three dimensions. It is true that most computer monitors really provide two-dimensional

representations of threedimensional objects, but that is irrelevant to 3D machining. Once the object exists in the computer at the CAD stage, it can be prepared for machining in much the same way that a 2½D object is dealt with at the CAM stage, by turning the 3D model into G code then machining the object. CAM software for doing this is much less common than for working in 2D and 2½D, but the principle is the same: take a file containing the spatial information about the object and use that to create G code. Then use the G code at the CNC stage to carry out machining operations.

Just as a 2D CAD program will generate a DXF file to be used at the CAM stage of the process, so a 3D program will generate an STL file (or an OBJ file or similar) that can be used at the CAM stage to produce G code for the mill.

There is another aspect to this, though. As CAD moves increasingly to the creation of 3D models, many objects are more easily designed in the 3D environment, even though their features are essentially two-dimensional. The output from a 3D design program could be a set of 2D drawings, and the features of the object could be reproduced in a 2D or 2½D program and then turned into G code, but that would be very inefficient. It is a question of how best to link the 3D design process directly to the CAM and CNC stages of production. One good way is to generate the 3D computer model and then move directly to a 3D CAM program that will produce the G code for the mill. This way, the workflow will suit the designer's thinking processes as well as the machining process. Two good tools for machining 3D objects are Vectric's Aspire and Cut3D CAD/CAM programs.

Aspire is the big brother of VCarve that can accept 3D information from STL files or can allow the designer to create a low-relief 3D model using built-in drawing tools. In either case, the output is G code for the CNC stage of the manufacturing process.

Cut3D is specifically designed to take an STL file and to create G code for the mill. Cut3D does not have the additional design capability of Aspire, but it can cope with objects having features on the underside and can also 'slice' a 3D model so that it can be

produced in sections for machining, then assembled into the physical 3D object.

The example shown in Figs 11-4 and 11-5 is of a life-sized model created by Greg St. George, based on an illustration created using SILO 3D, as one of the exercises in the book *3D Modeling in Silo: The Official Guide* by Anthony Ward, David Randall and Nevercenter. The exercise resulted in the creation of a computer-based virtual 3D model of a character called Jade Raven in the book, and Greg was so inspired by that character that he decided to give her life as a real physical 3D model he calls La Femme Rocketeer.

The project involved considerable additional preparatory work to scale, section and transform elements of the computer model before it was exported as an OBJ file then imported into Cut3D. Because of its size, the physical model was split into sections so that, for example, the boots are machined as two halves, one half for each side; and the head was split down the centre into right and left halves. Cut 3D was used to create ‘slices’ that could be machined separately and, in Fig. 11-5, the slices that make up the torso can be seen clearly where the shoulder will meet the arm.

Fig. 11-4 La Femme Rocketeer. (Photo: Greg St. George).

Fig. 11-5 Layered construction of the torso of La Femme Rocketeer. (Photo: Greg St. George).

After machining, the sections were assembled to produce the final physical model. Although this is a complex item, and not a project for the faint-hearted, the end result is a realistic and highly detailed life-size 3D model that really demonstrates the capabilities of software like Cut3D, as well as the process of moving from a virtual model to the real-life 3D equivalent. More details of Greg St. George's work can be seen at http://gregstgeorge.com.

3D machining paths tend to be complex and, as a result, 3D machining can take much more time than 2½D machining. One way to deal with this is to use a 2½D CAM package for the 2 or 2½D features of an object, then use a 3D CAM package for the remaining 3D features of the same object. Although this means running two (or more) CNC programs, one after the other, it can often produce a quicker and better result.

Some care is required over machining speeds and feeds and, in particular, the step-over settings used in order to produce a good surface finish on 3D work.

3D SCANNING

Some people like to make real models; and artists, sculptors and model makers may prefer modelling in clay, foam or plastic to make a maquette (a scale model) or a finished model. If that is to be the master for finished pieces, it needs to be transferred into a suitable format for a CAM or CAM/ CNC program to machine one or more finished workpieces.

This can be done by using a probe (Fig. 11-6) to digitally scan the object and transfer it to a 3D file in the computer as a 'point cloud', which is a series of points in space that represent points on the surface of the real object and corresponding 3D model.

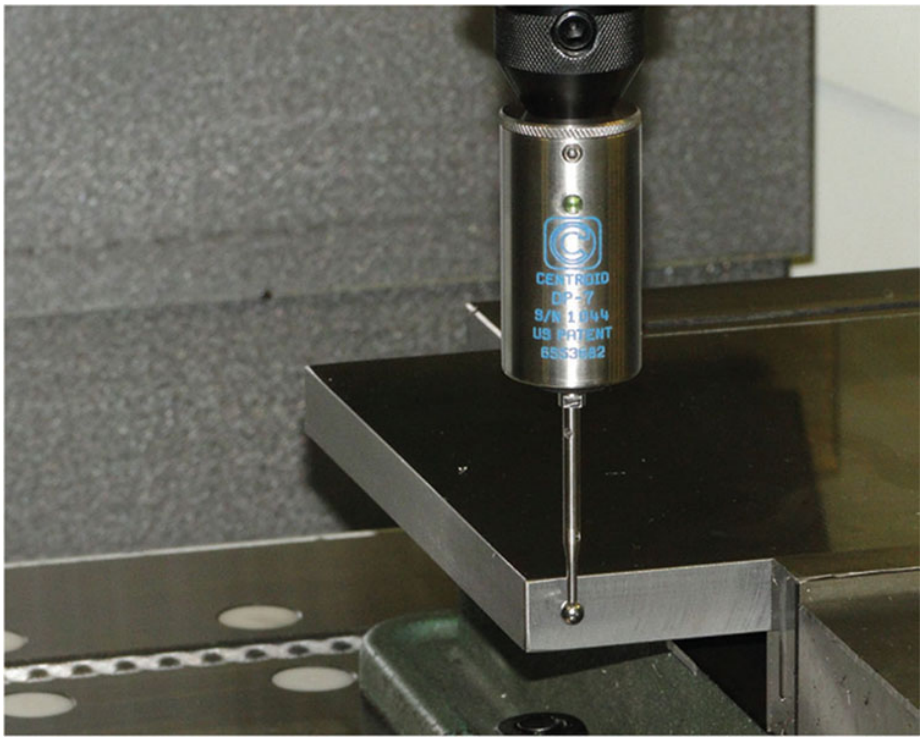


Fig. 11-6 Probe used for edge finding and for 3D probing. (Photo: Centroid Corporation).

Probes can be electromechanical or laser devices. The electromechanical probes are usually referred to as touch probes because they send a signal to the computer when they touch a

surface. If the touch probe is mounted in the spindle of a milling machine, and moved up and down as it is tracked backwards and forwards over the surface of an object on the mill table, the tip of the probe will be displaced slightly as it touches the object and that makes or breaks a circuit inside the probe and generates a signal. Software receives the signal and reads the position of the X, Y and Z axes, generating the coordinates of a point in space. Once enough points have been generated, the resultant point cloud represents the surface of the object. The closer the scanning points are to one another, the better the correspondence between the real object and the representation of the object in the point cloud data and the better the result when machining. This means that to reproduce fine detail, the scanning system must be capable of generating information about the coordinates of enough points to reconstruct the fine detail within the computer model. This depends not only on control of movement in X, Y and Z, but on the ability of the probe to respond as it touches the surface of the original object. Some features do not lend themselves well to being measured by some probes. Fig. 11-7 shows a situation where the probe requires a long arm for the tip to be able to reach the bottom of a hole, but that long arm may inadvertently contact a different point on the object, causing false tripping. The radius of the tip of the probe plays a part too, and a tip that is too large will make it impossible to detect fine detail (Fig. 11-8) or may falsely trip by catching on an adjacent edge as it descends towards a point on the surface (Fig. 11-9).

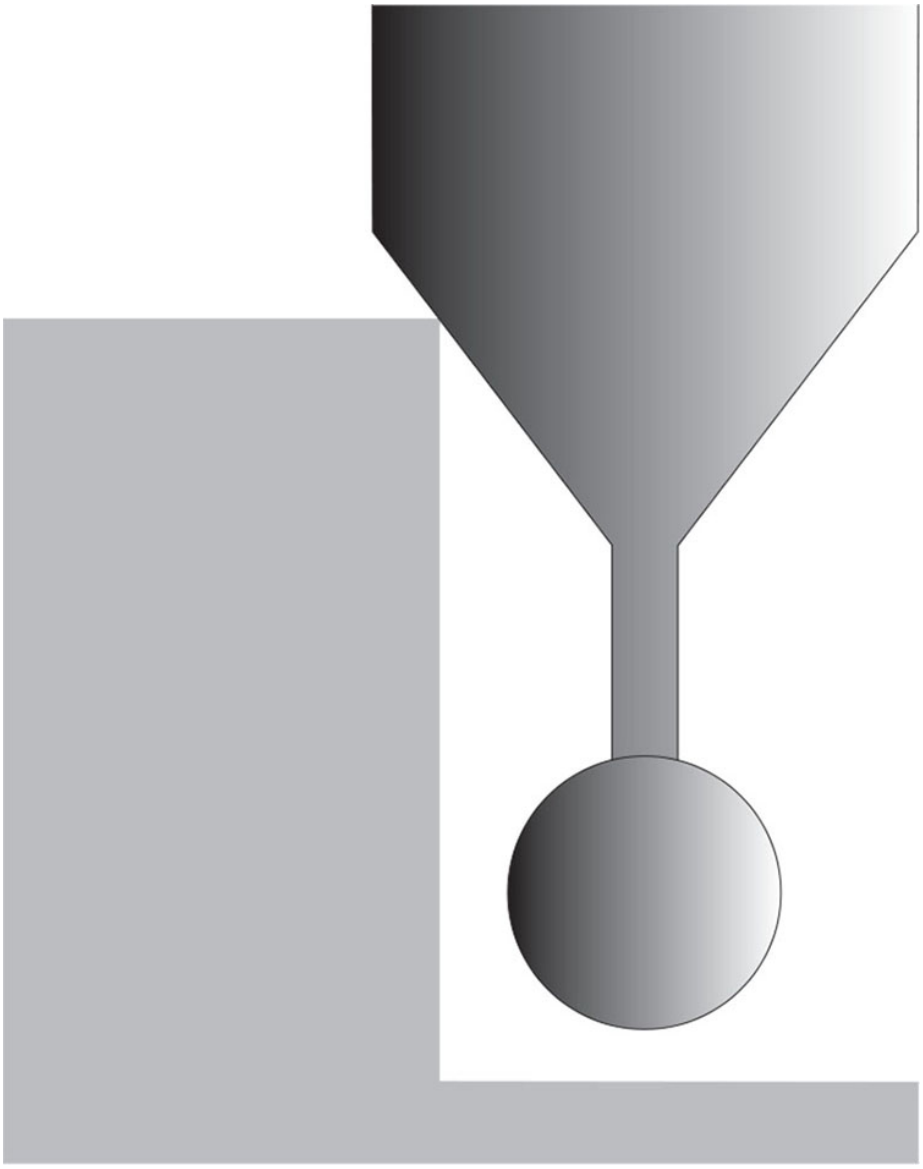


Fig. 11-7 Probe upper arm falsely tripping the probe.

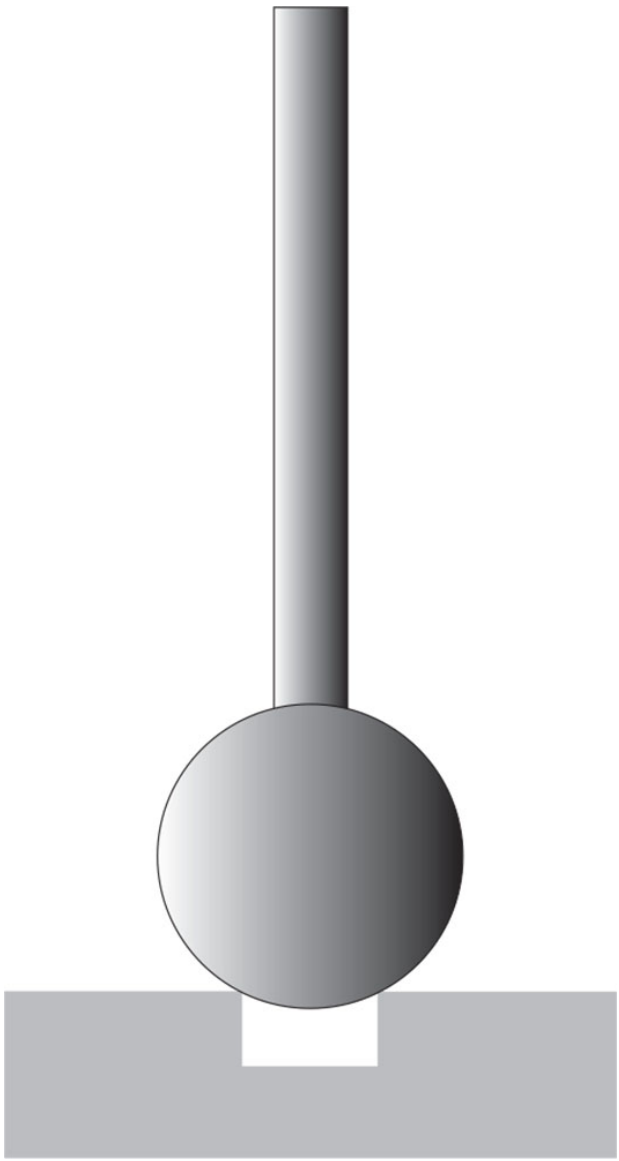


Fig. 11-8 If the tip of a probe is too large, it will not pick up small details.

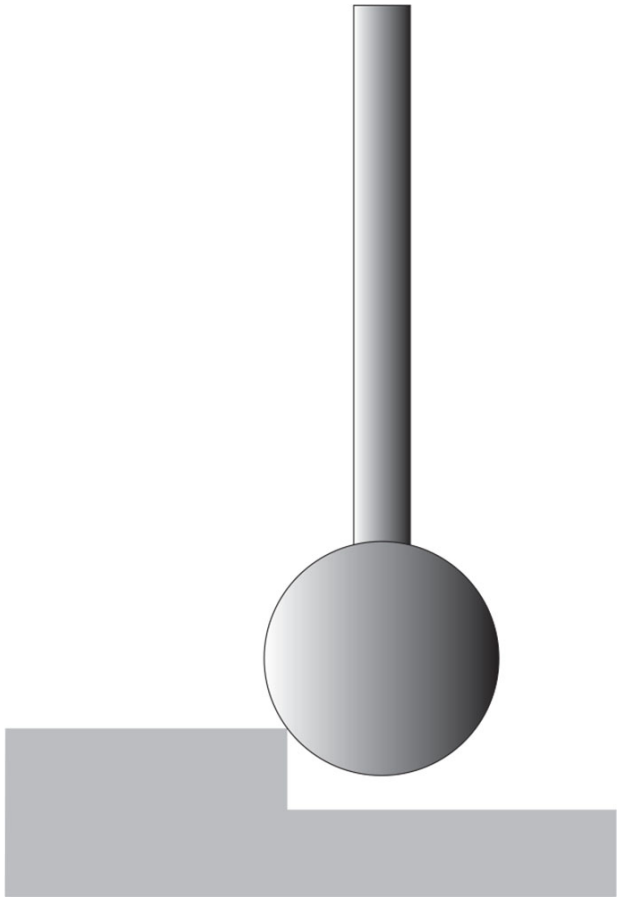


Fig. 11-9 A probe tip may trip on an adjacent edge, affecting its ability to accurately record detail.

Support for touch probes for scanning varies from one CNC program to another. Although neither LinuxCNC nor Mach3 have complete scanning functions built in, they both have G code commands that allow probing routines to be created. LinuxCNC uses **G38.2–G38.5** and Mach3 uses **G31**.

The basic method is straightforward. Begin by moving the probe towards a target point a little beyond where the probe is expected to trip. When the probe does trip, store the coordinates of that point. If the probe does not trip before reaching the target, there is an option to react to a ‘fault’ signal.

The most common uses of a probing movement are automatic tool height setting, finding an edge and finding the centre of a hole or

similar feature. Fig. 11-10 shows the basic method used to find the centre of a hole for the physical set-up shown in Figs 11-11 and 11-12. More accurate solutions are based on (a) discarding the first measurement and approaching the estimated tripping point more slowly to eliminate the effects of momentum and allow increased resolution with greater accuracy; and (b) taking more than one measurement and averaging the results.

Some solutions have been provided by users, and there are links to these on the support website. Be prepared to do some digging and some work to implement a solution to suit your own requirements.

Note that there is a significant difference in the software approach required when using a probe to scan in three dimensions, rather than to find an edge or a centre to set an origin in two dimensions. Finding edges and centres using a touch probe is a much simpler task, with simpler solutions. Probing in 3D normally results in many more points being identified by the probe, so these can be stored in a file. The sophistication in 3D probing techniques is in the prediction of appropriate target tripping points, the accuracy of the probe and what is done afterwards with the cloud of data points that result.

Software is usually available for commercial touch probes that are designed to scan objects, and that software will capture and store a point cloud in a straightforward manner. This is, however, a relatively expensive way to solve the problem.

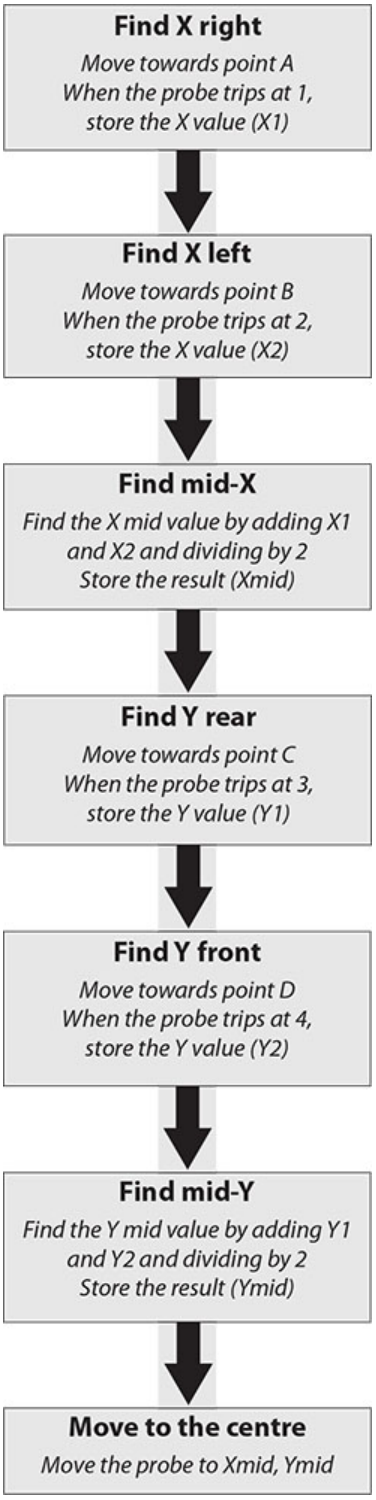


Fig. 11-10 Basic method for using a probe to find the centre of a hole.

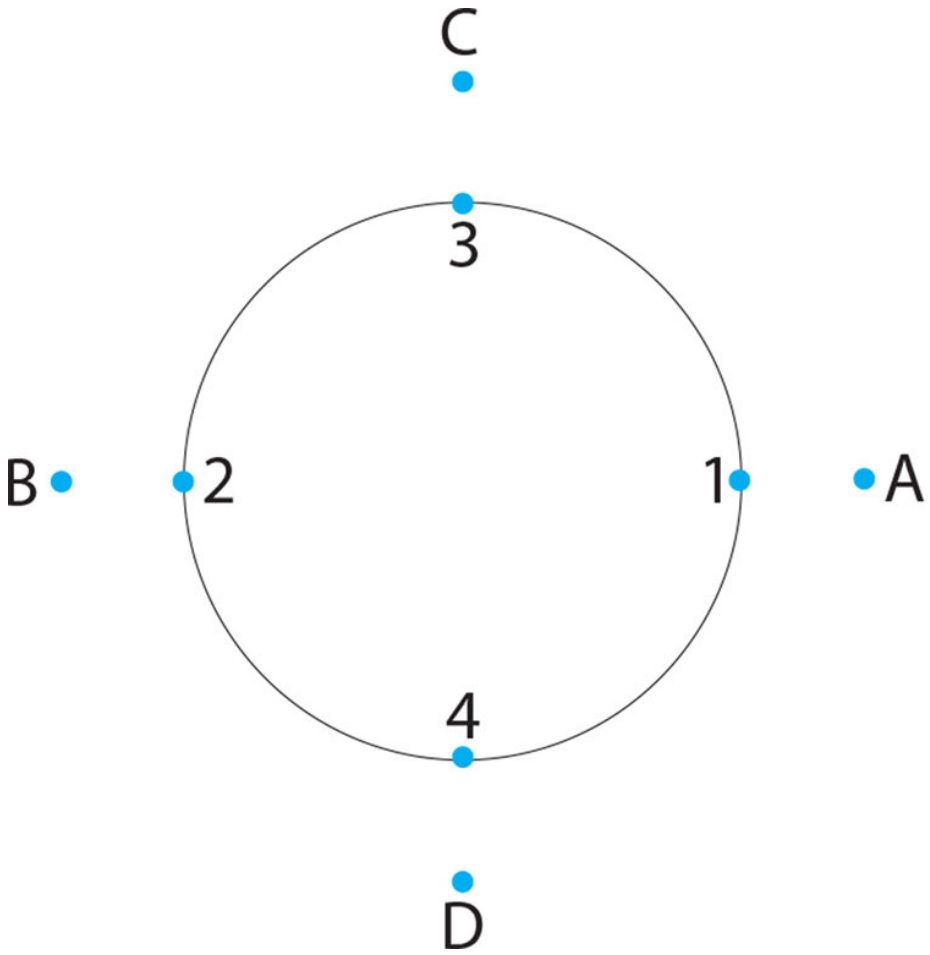


Fig. 11-11 Key points for the probing method outlined in Fig. 11-10.

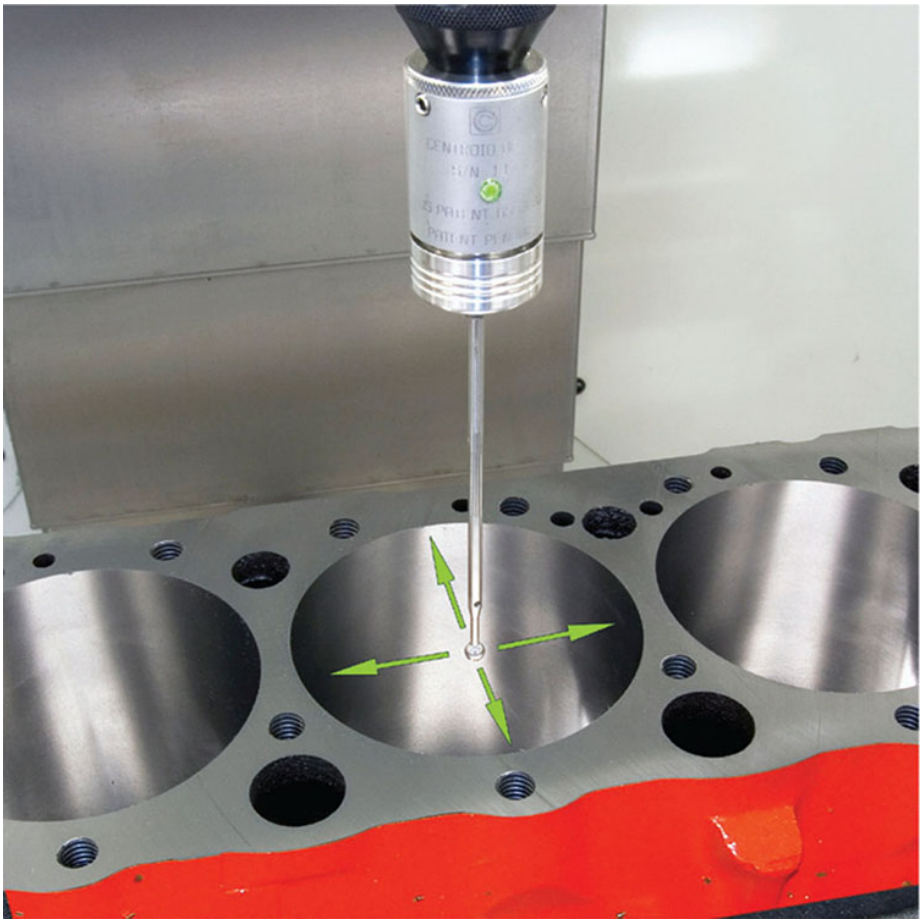


Fig. 11-12 Using a probe to find the centre of the bore in an engine cylinder block. (Photo: Centroid Corporation).

Fig. 11-13 shows a Centroid probe being used with Centroid software to generate data from the points on the surface of a workpiece. Fig. 11-14 shows the on-screen image generated by a scanning operation, the blue wax master produced from the data and the finished metal object cast from a mould using the blue wax master.

It seems likely that the current work being done in the field of low-cost 3D printers may result in the increased availability of accurate, low-cost 3D probes and scanning software that might easily be used to allow a milling machine to recreate a scanned object by machining.

Laser-based scanning systems are rather different and operate in a different way. Laser scanners are non-contact devices and do not need to touch the original object in order to capture information about the surface. Many laser probes use a range-finding technique where the light from the laser is beamed at an object and a range-finder measures the time taken for the reflected light to come back to the probe.

A different method is to use a laser and a camera to find the distance from the laser to the object by triangulation based on the position of the laser spot in the camera image. Triangulation laser scanners can produce very accurate measurements of distance for objects near the laser, while range-finding is more accurate at large distances.

There are several other variations on the ways in which laser scanners work, all with their own advantages and disadvantages. At present, there is little support available for laser scanners attached to milling machines, except for some commercially available probing systems.

Structured light scanners project a pattern of light on to the surface of an object, then use a camera to capture the image of the light pattern on the surface, and software to analyse the result, measuring the distortion of the pattern of light and calculating the coordinates of the illuminated points on the surface. These scanners often project a line of light on to an object and then calculate the coordinates of the points that lie under the line on the object.

At this stage in development, structured light scanners are best considered as standalone devices, generating data from an original object independently of a milling machine or the main CNC packages. The resulting data can be used as the basis for a 3D model of the object at the CAD stage of the CAD/CAM/CNC cycle.

Free software, available at www.davidlaserscanner.com, allows some experimentation with low-cost laser and structured light scanning. Commercial systems are increasingly available for both scanning methods.

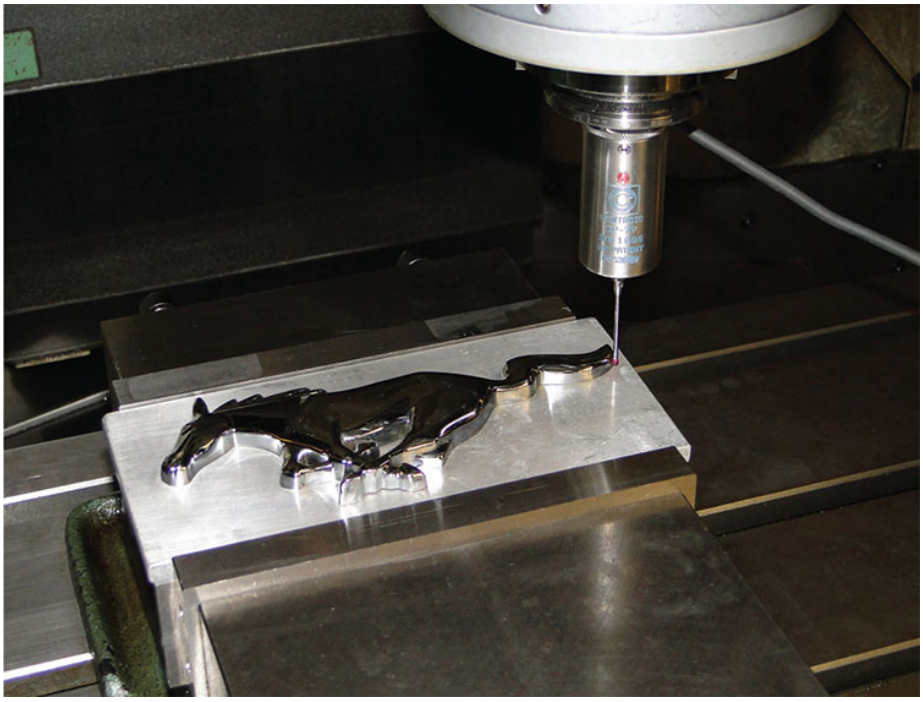


Fig. 11-13 Probing to construct a 3D model of a horse. (Photo: Centroid Corporation).

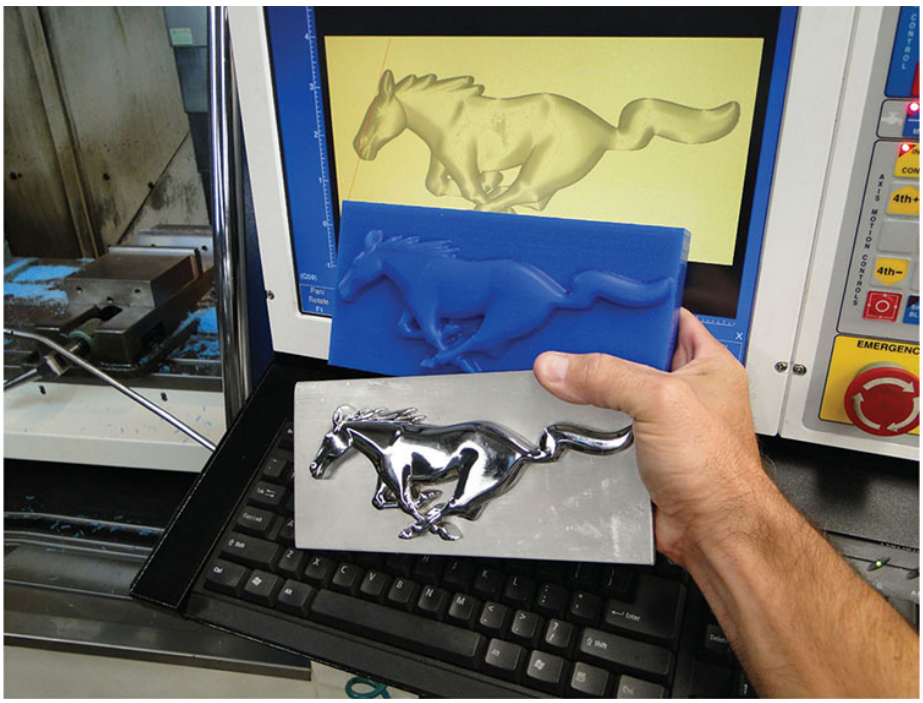


Fig. 11-14 The on-screen image of the 3D model of a horse; a wax master produced by CNC machining using cutter paths based on the 3D software model of the

horse; and the finished product made by casting from the wax master. (Photo: Centroid Corporation).

Once data has been generated by scanning, using whatever method is convenient or available, the Open Source software, MeshLab, can be used to process and edit the data. Data generated by scanning may contain noise or conflicting data, and some features on 3D models may require modification to produce the final shape for machining. MeshLab provides a number of useful tools for manipulating data, including slicing models, before outputting data in a file format suitable for importing into a CAD program to begin the CAD/CAM/CNC cycle.

MeshLab is a useful software tool developed with the support of the 3D-CoForum project and is available at http://meshlab.sourceforge.net/.

Commercial equivalents, such as Geo-magic, are also available.

It is always worth investing in good-quality safety equipment such as goggles, ear defenders and gloves.

Appendix I

HEALTH AND SAFETY

Machinery is inherently dangerous, with sharp edges and moving parts. Machinery moving under computer control brings particular hazards with loss of control, unexpected movements, and drive motors operating slides with a more powerful action than a human can stop.

In an industrial place of work, there are clear regulations to protect everyone, and rightly so. In the hobbyist's workshop, where machinery is used for recreational purposes, there can be additional dangers just because there is no compulsion to protect oneself or others. Industrial Health and Safety legislation does not apply at home and the Health and Safety Inspectorate does not make home visits. This can lead to a lax attitude, either because the regulations do not apply or because, for many hobbyists, there is little concept of the requirements that would apply at work. All of this leads to additional dangers.

Machinery is no respecter of persons so, whatever the circumstances and whatever the location, we would do well to take care and to follow the kinds of safe working practices expected in industry. They have been developed for good reasons.

Safety guards are the butt of many jokes and are often discarded. They are there for a purpose. Many CNC machines are home-constructed or home-modified, and the last thing that is tackled (and the main thing that is forgotten) is usually the safety guards around moving parts like the spindle and the cutting tools. Protect yourself with a guard that prevents you from touching the rotating tool. Sometimes this is best done with a large guard or cabinet that

encloses the machine when it is running. That way, the set-up of tool and job is easier and the tool remains unobstructed as the machine moves. If it is easy to use and does a useful job, it is more likely to be used. If it is interlocked to the spindle motor so that the motor will not operate until the guard is in position, so much the better. Yes, it takes longer to get to the point at which the machine is cutting, but what is a few seconds when a finger is at stake?

You would not climb a razor-wire fence with your bare hands, so do not touch swarf with your bare hands. Not ever.

Do not go anywhere near a rotating tool that is cutting material. Do not try removing swarf from around a revolving cutter. Not ever.

Do not put your face near a revolving cutter 'just to get a better look at what the tool is doing'. Your cheeks may be weather-beaten, wizened and as hard as leather, but that only makes it more difficult for a plastic surgeon to match the skin tone.

Protect your eyes and ears at all costs.

Wear good eye protection made to proper industrial standards at all times, even if you wear spectacles. Proper wraparound, ski-style industrial goggles with impact-resistant lens material and soft seals around the face allow the comfortable use of spectacles. Eye protection should be able to resist puncture and impact. It should be comfortable to wear and when it gets scratched, you should replace it. Think of it as a cheap investment in the most sensitive biological devices imaginable. Eyes cannot be replaced and good protection is cheaper than medical bills.

Wear good ear protection whenever your machine is running. As with eye protection, buy ear protectors made to industrial standards. You might prefer the cheapest foam earplugs the chemist will sell, but a better buy is certified, full-enclosure earmuffs that provide at least 30dB attenuation. You will find that reducing your exposure to noise also reduces the fatiguing effect on the brain. High-speed spindles create astonishingly high levels of sound, full of very damaging, high-frequency tones that will dull hearing very quickly indeed. That hearing loss is permanent and cannot be restored.

Put two hooks somewhere near your machine: one for the goggles and the other for earmuffs. It is more likely that you will use

these if they are handy.

Take a moment to assess the risks whenever you are about to use machinery. It does not take long to run through what you expect to happen, and what might go wrong, and take steps to ensure your own safety before you operate machinery. It is a good idea with hand tools too. Risk assessment is a simple procedure that you should get into the habit of practising, for your own safety. It can prevent damage to your machinery as well, and that is always a good thing.

Be aware of the hazards of cutting fluids and oils. Read the labels and avoid undue exposure. A good barrier cream can be useful for protection from some substances, as can decent disposable gloves.

Take particular care when someone else is in the workshop. It is always delightful to show off one's machinery. Let it be a safe as well as a joyful experience. You do have a duty of care to visitors. And they may have good lawyers.

Throughout this book, there are pictures of cutters and machinery that do not always have guards visible. These pictures are for illustration, to help you to see and understand what is being said or shown. Nothing in this book should be taken as an indication that we suggest, recommend or endorse working practices that are potentially hazardous or unsafe.

Safe working practices will help ensure the workshop is a place of relaxation and enjoyment.

Appendix II

G CODES, M CODES AND OTHER CODES

Beige denotes commands available only in LinuxCNC
Pink denotes commands available only in Mach3

This table is for information and guidance and may be used as a quick reference. It is essential that you consult the manual for the CNC software you are using to control your mill, to find out exactly how to use these commands to control your mill.

G Codes

G code	Description
G0	Straight line motion at rapid speed
G1	Straight line motion at Feed Rate (as set by the F word)
G2	Clockwise circular or helical motion at Feed Rate
G3	Anti-clockwise circular or helical motion at Feed Rate
G4	Dwell (i.e. Pause)
G5.1	Quadratic B-Spline
G5.2	NURB s Block
G10 L1	Set Tool Table Entry
G10 L10	Set Tool Table, Calculated, Workpiece
G10	Set Tool Table, Calculated, Fixture

L11	
G10	Coordinate System Origin Setting
L2	
G10	Coordinate System Origin Setting Calculated
L20	
G12	Clockwise circular pocket
G13	Anti-clockwise circular pocket
G17	Select XY plane
G17.1	Select UV plane
G18	Select XZ plane
G18.1	Select WU plane
G19	Select YZ plane
G19.1	Select VW plane
G20	Unit of measurement : inches
G21	Unit of measurement : millimetres
G28	Rapid move from current position to the absolute position stored in parameters 5161-5166, via an (optional) intermediate position.
G28.1	Store the absolute co-ordinates of the current position in parameters 5161-5166
G28.1	Reference (Home) the named axes
G30	Rapid move from current position to the absolute position stored in parameters 5181-5186, via an (optional) intermediate position.
G30.1	Store the absolute co-ordinates of the current position in parameters 5181-5186
G31	Straight probe
G33	Synchronized motion of axes and spindle
G33.1	Rigid tapping
G38.2	Probing
-	
G38.5	
G40	Turn cutter radius compensation off
G41	Turn cutter radius compensation on (left)

G41.1	Dynamic cutter radius compensation (left)
G42	Turn cutter radius compensation on (right)
G42.1	Dynamic cutter radius compensation (right)
G43	Use tool length offset from tool table
G43.1	Dynamic tool length offset
G49	Cancel tool length offset
G50	Set scale factor to 1
G51	Define and apply a scale factor
G52	Temporarily offset the current co-ordinate system
G53	Motion in Machine Coordinate System (absolute co-ordinates). This is a non-modal command which does not persist beyond the line on which it is used.
G54-G59	Select Work Offset co-ordinate system (1 - 6)
G59	Select Work Offset co-ordinate system (1 – 254)
G59.1-G59.3	Select co-ordinate System (7 - 9)
G61	Exact path control mode
G61.1	Exact stop path control mode (same as G61)
G61	Exact stop path control mode
G64	Constant velocity path control mode
G64	Best speed path control mode with optional tolerance
G68	Rotate the current co-ordinate system
G69	Cancel rotation of current co-ordinate system
G70	Unit of measurement : inches (Note: G20 is the preferred command for this)
G71	Unit of measurement : millimeters (Note: G21 is the preferred command for this)
G73	Canned cycle: High speed peck drill
G80	Cancel canned cycle motion modes
G81	Canned cycle: drilling
G82	Canned cycle: drilling with dwell
G83	Canned cycle: drilling with peck

G85	Canned cycle: boring with no dwell, and feed out at feed rate
G86	Canned cycle: boring with spindle stop at the bottom, and feed out at rapid feed rate
G89	Canned cycle: boring with dwell at the bottom, and feed out at feed rate
G90	Absolute distance mode
G90.1	Absolute IJ mode for G2 and G3 commands
G91	Incremental distance mode
G91.1	Incremental IJ mode for G2 and G3 commands
G92	Offset current co-ordinate system and store offsets
G92.1	Cancel G92 offsets and erase stored offsets
G92.2	Cancel G92 offsets without erasing stored offsets
G92.3	Set G92 offsets to stored offset values
G93	Set feed to: inverse time mode
G94	Set feed to: units per minute mode
G95	Set feed to: units per revolution mode
G96	Spindle: constant surface speed mode
G97	Spindle: RPM mode
G98	Canned cycle Z retract mode: retract to Z value used before cycle began
G99	Canned cycle Z retract mode: retract to Z value specified within the canned cycle command

M Codes

Code	Description
M0	Program pause (also see M60)
M1	Program pause if optional Stop setting is ON
M2	Program end (also see M30)
M3	Start spindle rotating clockwise
M4	Start spindle rotating anti-clockwise
M5	Stop spindle

M6	Tool change
M7	Turn mist coolant on
M8	Turn flood coolant on
M9	Turn coolant off
M30	Program end and rewind to start of program. LinuxCNC exchanges pallet shuttles (if they exist) before ending the program then rewinding.
M47	Repeat program from first line
M48	Enable feed and spindle speed overrides
M49	Disable feed and spindle speed overrides
M50	Enable/disable feed override
M51	Enable/disable spindle speed override
M52	Enable/disable adaptive feed
M53	Enable/disable feed stop switch
M60	Exchange pallet shuttles then pause the program
M61	Set current tool number within manual mode or MDI mode
M62	Enable digital output synchronized with motion
M63	Disable digital output synchronized with motion
M64	Enable immediate digital output
M65	Disable immediate digital output
M66	Wait for input
M67	Set an analogue output synchronized with motion
M68	Set an immediate analogue output
M98	Call subroutine
M99	Return from subroutine. Also used to repeat a program from the beginning (same as M47).
M100-M199	User Defined M-Codes

Other Codes

Code	Description
F	Set feed rate
S	Set spindle speed
T	Select tool
O	Control of the logical flow through the program (subroutines, loops, conditional branching, repeat, and indirection).

Appendix III

INITIALIZATION BLOCK

An initialization block is often used at the start of a G code program to set the machine into a known state in preparation for the program instructions that follow. For convenience, this appendix contains the initialization block mentioned in this book.

In cases where the machine is stopped unexpectedly, or where it seems to be behaving strangely in response to commands, typing these commands in MDI mode will often return the machine to normal behaviour. The initialization block is not the only thing that affects the behaviour of the machine, so in difficult cases you may also have to examine any settings that have changed in the set-up or INI files.

In the light of experience, you may wish to develop your own initialization block to suit your own approach and your own machine, but the commands below set the obvious aspects of machine behaviour to a known, usable state.

Instructions for millimetres	**Alternative instructions for inches**
G21 (set units to millimetres)	G20 (set units to inches)
G90 (use absolute distances)	
G91.1 (incremental arc mode)	
G94 (feed per minute mode)	
G92.1 (cancel offsets)	
G54 (use coordinate system 1)	

G98 (retract behaviour for canned cycles)
G49 (cancel tool length offset)
G40 (cancel cutter compensation)
G17 (select XY plane)
G80 (cancel canned cycle mode)

Further Information

There is a support website for this book at www.cncintheworkshop.com that includes additional content and links to sources of support.

There is lots of support for the software packages mentioned in this book. These include:

- Mach3 www.machsupport.com and specific support forums on CNCZone (in Machine Controllers, Software and Solutions) and Yahoo! Groups;
- LinuxCNC www.linuxCNC.org and specific support forums on CNCZone (in Machine Controllers, Software and Solutions);
- the main LinuxCNC mailing list – provided via Sourceforge and at https://lists.sourceforge.net/lists/listinfo/emc-users;
- Vectric software – Cut2D, Cut3D, VCarve Pro, Aspire and PhotoVCarve at www.Vectric.com including a collection of training videos, and support forums for each package;
- forums also on www.CNCZone.com www.signforums.com www.microcarve.com www.worldofwoodforum.com and several others.

Artists and manufacturers who supplied information or photographs can be found at:

- Allendale Electronics Ltd www.machine-dro.co.uk
- Arc Euro Trade Ltd www.arceurotrade.co.uk
- Automation Components Ltd www.automationcomponents.co.uk
- Centroid Corporation www.centroidcnc.com

- Enseignes Bois et Passions www.enseignesbp.com or the English version at www.woodsigns.ca
- Greg St. George http://gregstgeorge.com
- Laser Center/Edge Finder (SDA Manufacturing) www.lasercenteredgefinder.com
- Pictures in Wood (Markus Dahlberg) www.pictures-in-wood.se
- Sherline Products Inc www.sherline.com
- Tormach LLC www.tormach.com
- Shopbot Tools Inc www.shopbottools.com;
- Stritzelberger Steuerungstechnik GmbH www.vakuumtisch.de
- Vectric Ltd www.vectric.com

Index

Index of Commands