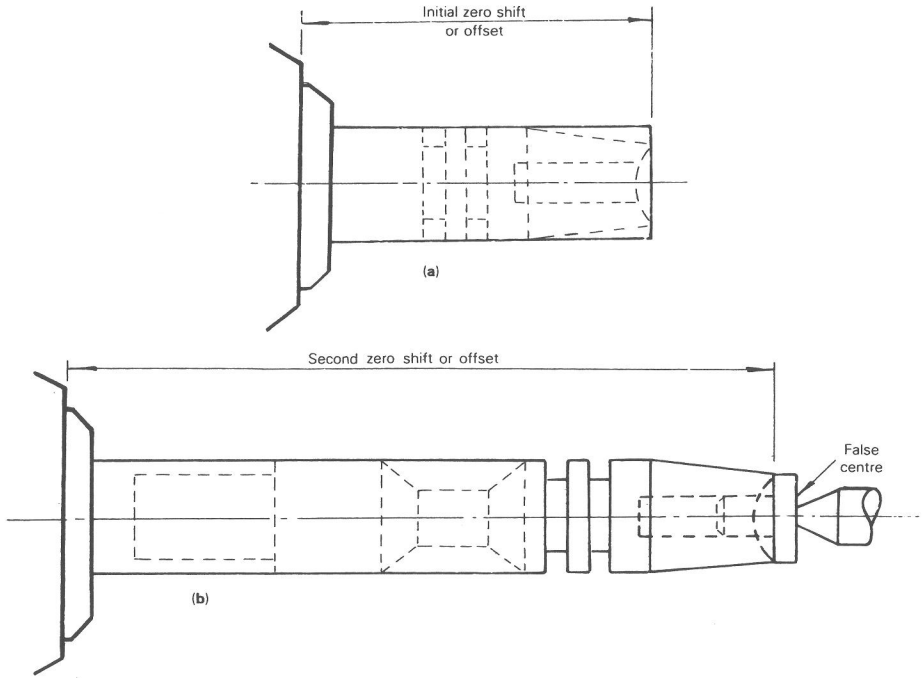


Figure 8.7 *The application of a second zero shift to accommodate work resetting* (a) *first work setting* (b) *second work setting.*

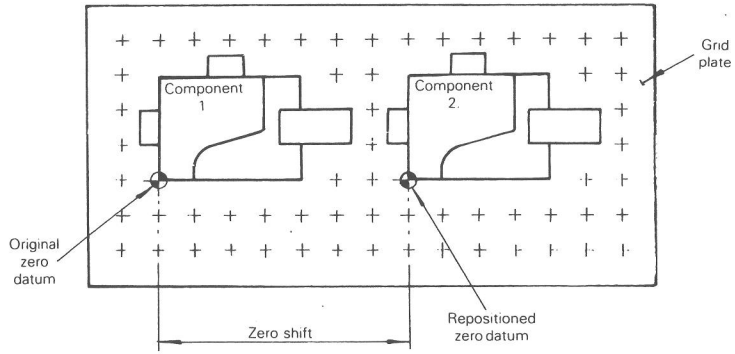


Figure 8.8 *Use of zero shift for multicomponent machining at one setting.*

to locate a particular town. Using this reference system the part programmer can instruct the machine set-up person/operator exactly where to position each component so that their location will correspond with the selected program zeros.

DOCUMENTATION ASSOCIATED WITH PART PROGRAMMING

Before a part program can be compiled it is necessary to give some thought to the practical aspects of producing the component, and in most companies this is likely to involve the completion of an operation sheet. There is no standard operation sheet, and the format will vary from company to company. One which will meet the requirements of the exercises that follow is shown in Figure 8.9.

In addition to an operation sheet there is also the need for documentation relating to machine setting and tooling, because some of the decisions made during the operation planning stage, and which in turn are taken into account when writing the part program, are of direct concern to shopfloor personnel responsible for preparing the tooling and the machine. Again, there is no standard format for such documents. Each company will have its own procedure.

DOCUMENTATION RELATING TO MACHINE SETTING

Information regarding work-holding and location is of vital importance to the machine set-up person. He or she will also benefit from knowing the sequence of operations that has been adopted by the programmer. Also, it will be necessary to know the form in which the material to be machined is to be supplied. Ideally, all this information should be documented, not only as an aid to efficiency on the shop floor, but also to provide a record for future reference.

The documents used to convey this information will vary from company to company, and the precise way this information is disseminated is not of major

OPERATION SCHEDULE		PART No.		DESCRIPTION		SHEET No OF
		MACHINE TYPE		COMPILED BY		DATE
OP No.	DESCRIPTION	TOOLING TYPE AND SIZE	WORK HOLDING	CUTTING SPEED	FEED RATE	SPINDLE SPEED

Figure 8.9 *Example of an operation schedule.*

importance. The important thing is that the shop floor personnel fully understand what is required. So how detailed does the information need to be? The answer depends on the complexity of the component and the machining operations involved.

Assume that the machine set-up person knows the sequence of machining operations involved and is to proceed with setting up the machine. Consider in the first instance work loading, holding, and location. What information is required?

A simple component that is to be turned in one set up from a prefaced billet could be accommodated with a few short notes as follows:

Material:	prepared billet, part number ****
Loading:	manual
Work-holding:	chuck type, fixture number ****
Location:	back face of chuck
Zero shift:	*Z* direction + or −, and value

The last item would indicate that a manual data entry shifting the *Z* axis zero from the spindle face to the workpiece face is required.

A more complex component requiring two settings, with the second operation requiring center support activated by an entry in the program, will require a little more detail and the information may be given as follows:

Material:	diameter and length of bar stock
Loading:	bar feed to programmed stops, bar stop number
Work-holding:	collet, with programmed center support for second setting
Zero shifts:	first setting, direction + or −, and value second setting, direction + or −, and value

This information could be supplemented by two simple sketches showing the machining to be carried out at each setting.

A simlar exercise can be carried out for workpieces involving milling. The exercise shown in Figure 8.10 could be produced on a "one part" basis or involve a multicomponent setting.

In the first instance the workpiece could be located using the corner of the fixed jaw of the vise as a reference point, a technique referred to on page 168. The instructions necessary to achieve this would be as follows:

Material:	prepared blank length × width × height
Work-holding:	machine vise, fixture number ****
Location:	left-corner of fixed jaw
Program datum:	*X* axis −25 mm (−1 in.) (axis-direction + or − value) *Y* axis 25 mm (1 in.) *Z* axis 2 mm (0.1 in.)

Again the information regarding the program datum may be more readily understood if the instructions include a sketch.

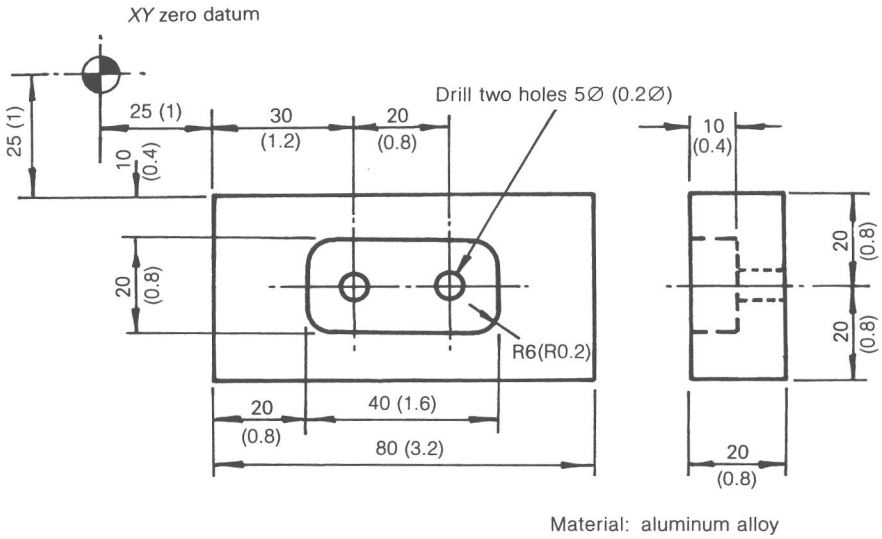


Figure 8.10 *Component detail. (Inch units are given in parentheses.)*

A multicomponent setup involving the same component could involve the use of a grid plate. To convey the necessary set-up information, the programmer should be familiar with the grid plate and its associated locating and clamping devices. With such knowledge he or she may be able to give detailed instructions for the complete setting, using the grid references to position the various setting blocks, locating dowels, and clamps to be used in the operation. On the other hand, a competent set-up person could manage with the basic information included in Figure 8.11.

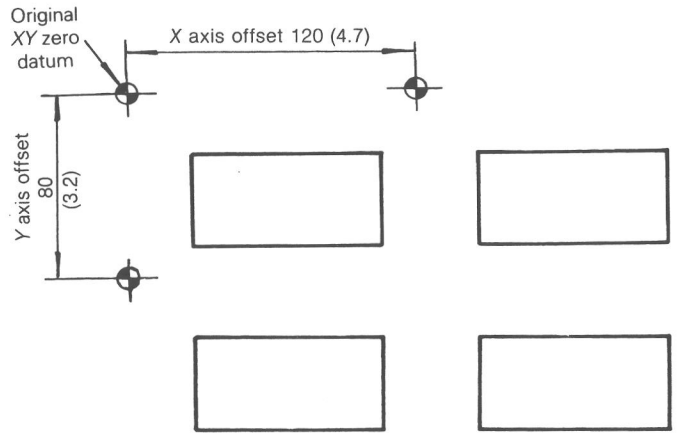


Figure 8.11 *Use of grid plate. (Inch units are given in parentheses.)*

TOOLING SELECTION AND IDENTIFICATION

The responsibilities of the part programmer concerning tooling are as follows:

(a) determine the appropriate tools to be used, including their shape and size and the material from which they will be made;
(b) allocate identification numbers to facilitate machine setting;
(c) allocate tool offset numbers;
(d) determine, when appropriate, the dimensonal value of the offsets;
(e) prepare appropriate documentation.

It is essential that a programmer is fully conversant with the tooling system for the machine involved, that is, the type of tooling that can be used and the way the tools can be located and held in position.

A major feature of CNC machining is the use of standard tooling. The intricate slide movements that are possible greatly minimize the need for special tooling, particularly form tools. In many ways the tooling requirements for CNC machining are less complex than for conventional machining.

Providing the programmer is conversant with the machine tooling system, the process of selecting tooling for a particular job is largely a case of selecting and utilizing standard items.

It is important that the correct tool material is used, particularly when using carbide inserts. Reference should be made to manufacturers' literature for guidance in this respect. Pages 44 through 46 give an indication of the type of information that is available.

It is often the case that the tools available within a company for use on a particular machine will be further standardized with their details being documented. An example of a company-based tool standard is shown in Figure 8.12.

All tools are required to have a numerical identity within the part program. This identity, commonly the letter T followed by two digits, is allocated by the part programmer and will correspond with the numbered position the tool will occupy in the machine turret, magazine, or other storage facility. The position each tool will occupy is affected by factors which are discussed below.

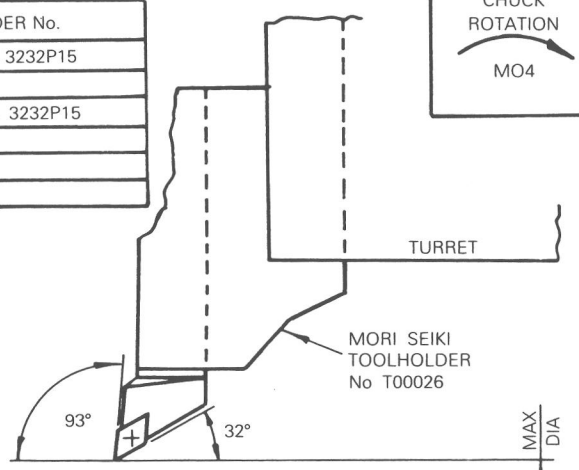
Commonly used tools are often given an identity that is retained at all times, since this often eliminates the need to reset when jobs are changed. When this situation exists, it is essential that the part programmer knows exactly which tools are involved and their numerical identity.

TOOL STORAGE

With automatic tool changing facilities involving turrets, the positions for the tools in the turret are numbered. Thus a tool call of, say, T06 will cause the turret to index to position number six. The tool allocated the numerical identity 6 must be set in position six.

A.J. SMITH LTD — NUMERICAL CONTROL DEPT.

TOOL HOLDER No.	
ISO CODING	PDJNL 3232P15
KENNAMETAL	
SANDVIK	PDJNL 3232P15
VALENITE	
CARBOLOY	



NOTE:- ALL DIMENSIONS ARE IN INCHES UNLESS STATED OTHERWISE

TOOL No	RN	AC	AN	XN	ZN	WN	AS	XS	ZS
T709									
T710	0·008	3	55	0·008	0·008		-90	3·5634	2·6795
T711	0·0156	3	55	0·0156	0·0156		-90	3·5568	2·6788
T712	0·0313	3	55	0·0313	0·0313		-90	3·5433	2·6772
T713	0·0469	3	55	0·0469	0·0469		-90	3·5297	2·6756
T714	0·0·625	3	55	0·0625	0·0625		-90	3·5162	2·6741
T715	0·0937	3	55	0·0937	0·0937		-90	3·4892	2·6710
T716									

NOSE RADIUS	INSERT No. I.S.O. CODE	I.S.O. CODE	I.S.O. CODE	I.S.O. CODE	Zero Rtn. Dia.	MAX. DIA.
0·008	DNMG150602				20·4323	25·9441
0·0156	DNMG150604	DNMM150604			20·4455	25·9573
0·0313	DNMG150608	DNMM150608			20·4725	25·9843
0·0469	DNMG150612	DNMM150612			20·4997	26·0115
0·0625	DNMG150616	DNMM150616			20·5267	26·0385
0·0937	DNMG150624				20·580.7	26·0925

TITLE:- FINISH PROFILING TOOL	TOOL No
QUALIFIED TOOLING MORI SEIKI SL7 — C.N.C. LATHE	T709 to T716

Figure 8.12 *Company-devised tool standard. (All units are given in millimeters.)*

Similarly, tools changed by automatic handling devices will be housed in readiness in a tooling magazine. When a tool is called the magazine will index to bring the appropriate tooling station into a position where the tool located in that station can be accessed by the handling device. Clearly the correct tool must be in each numbered position if the programmed tool call is to bring the desired tool into the machining position.

Even when the tool change is a manual operation, effected by a programmed stop in the machining cycle, the process is assisted if the operator has a clear indication of the next tool to be used. It is usual, therefore, to number the tool storage positions or even the tools themselves. When the programmed break in machining occurs, the operator can refer to a document provided by the programmer to determine the next tool involved; on the more sophisticated control systems the tool may be indicated by a message displayed on the visual display unit of the control.

The programmer should give due thought to the positioning of the tools in relationship to each other in the turret or magazine. Most indexing arrangements involve rotation in one direction only, so to change, say, from T03 to T06 will require three indexing moves, two of which are time-consuming and unproductive. Therefore the objective should be to position the tools in the turret or magazine in the order in which they will be called into use, although this is not always possible in practice.

The problem of wasteful indexing time is considerably eased when the machine is equipped with the facility to index tooling by the shortest possible route. In other words, the turret or magazine will rotate either clockwise or counterclockwise depending on which tool is called.

TOOL CHANGING POSITION

The programmer should consider carefully the position the machine slides are to be in when a tool change is made. There is a tendency, particularly among students, to return the machine slides to a set position before making a change, a practice that may have its merits from a safety point of view early in training but which, like wasteful indexing moves, can add considerably to the total time taken to machine the part.

The objective must be to keep noncutting slide movement to a minimum. For example, on a vertical machining center it is often possible to effect a tool change immediately above the point at which the tool completes the required machining, the change being carried out after an appropriate Z up-movement of the machine spindle or head. This saves making a long and unnecessary journey to a set position such as the XY zero datum. On turning centers a similar time saving can be achieved by indexing as near to the workpiece as is safely possible. The programmer should always refer to the machinery manuals to check clearances necessary to allow tool indexing mechanisms and cutting tools to clear any obstructions.

REPLACEMENT TOOLING

For long production runs the programmer will need to give some thought to the provision of replacement tooling.

When tools need to be replaced it is possible for the set-up person to determine suitable offsets and make the necessary tool data entries as he or she would for the original tools, but this is time-consuming and interrupts production.

An alternative approach is to use replacement tooling which is identical to the original. Such identical tooling may be of two types, namely, "qualified" or "preset."

Qualified tooling is used on turning centers and has dimensions guaranteed by the manufacturer to within ± 0.0005 in. or ± 0.08 mm from up to three datum faces.

Preset tooling is precisely set to predetermined dimensions in the toolroom and is applied to turning tools and milling cutters.

The programmer may choose to recommend qualified or preset tools when compiling his or her tooling schedule, but if such tooling is prescribed, the programmer may need a feedback of information from the toolroom regarding the setting sizes. This information then becomes part of the overall programming and machine-setting package and should be documented for future reference.

TOOLING DOCUMENTATION

Documentation regarding tooling, as with machine setting instructions, may be simple or relatively complex. It depends largely on the size of the company and the degree of organization that exists.

The possibilities range from the situation where the machine set-up person has personal access to the range of tooling likely to be required, to situations where the tooling is prepared in a special-purpose tool room, issued to the set-up person as a package for that particular job, and on completion returned to the tool room for refurbishment and storage.

For each programmed tool the minimum information required on the shop floor is as follows:

(a) programmed identity—T01, T02, T03, etc.;
(b) tool type;
(c) holder type and size;
(d) insert type and size;
(e) overall dimensions (solid tools);
(f) projection of cutting tool from holder.

When presetting is involved, the tool design or program personnel usually determine the original preset dimensions. The sizes should ultimately be no-

tified to the part programmer so that they may be recorded and included as part of the general documentation for that particular job. A well-organized tool preparation facility may well retain the data against their own job reference to facilitate the preparation of replacement tooling and to allow for the possibility of having to prepare identical tools at some future time. See Figures 8.13 and 8.14 for examples of tool data sheets.

When tooling offsets are being used to achieve a particular machining effect, as discussed on page 225, the value of the offsets must be included on the document.

It is often the situation that information regarding tooling, and sometimes information relating to machine setting, is included on the original part program form when one is used. Information documented in this way is of necessity rather brief, but in many cases is adequate.

Another practice widely adopted is to give tooling details alongside the tool call in the part program. Again, the information is brief but adequate for many situations.

The important thing is that the part programmer fully appreciates the needs of the people more directly concerned with the machining operation. There must be an efficient transfer of the relevant information. The means adopted

MODEL 104 TOOL SHEET

PART NO. 001-001	MATERIAL 1018 CRS	OPERATION 3	PROGRAMMED BY G. COMBS
PART NAME SAMPLE	B/P CHANGE DATE		DATE 5-6-80 SHEET 1 OF 1

TOOL SEQ.	TOOL NUMBER	TOOL DESCRIPTION	TOOL DIAMETER PROG.	TOOL DIAMETER ACT.	#	TOOL LENGTH PROG.	TOOL LENGTH ACT.	*	OPERATION DESCRIPTION	SPEED R P M	FEED IN/MIN
1	T01	HHS END MILL - 4 FLUTE ¾" SHANK SINGLE END/CV-49-15920 END MILL			D1 .125			H1	FINISH MILL PERIPHERY		
	E1	HOLDER	1.0	.750		6.187	5.812			270	3
2	T02	HSS END MILL - 4 FLUTE 3/8" SHANK DBLE END/CV-49-15915			D2			H2	Mill (4) pockets .062 Dp. Finish mill sides		
	E1	END MILL HOLDER	.250	.250		4.687	4.687			1100	2
3	T03	#2 CENTER DRILL Erikson Ext Collet Chuck/#200-3/16 collet/CV-49-15923							SPOT (18) HOLES ON PERIMETER, (8) ON B.C. and (4) on angle.		
	E1	Collet									
		Holder/#100-3/4 collet						H3 .125	.1 DP		
			.078	.078		4.875	5.000			1180	3
4	T04	Drill CV-49-15923 collet holder/#100-1/8 collet						H4 -.250	Drill (8) HOLES ON B.C.		
	E1		.125	.125		5.125	4.875			3000	6
5	T05	DRILL CV-49-15923 collet holder/with #100-7/32 collet						H5 .250	DRILL (18) HOLES ON PERIMETER AND (4) ON ANGLE 1" THRU		
	E1		.205	.205		5.250	5.500			1650	5
6	T06	HSS END MILL - 2 FLUTE 3/8" SHANK DBL END/CV-49-15915 END MILL HOLDER						H6 .138	C'BORE (18) HOLES		
	E1		.375	.375		4.562	4.700			850	3.4

Figure 8.13 *Model 104 tool sheet. Ex-Cell-O Corp., Rockford Machine Tool Company, Rockford, IL*

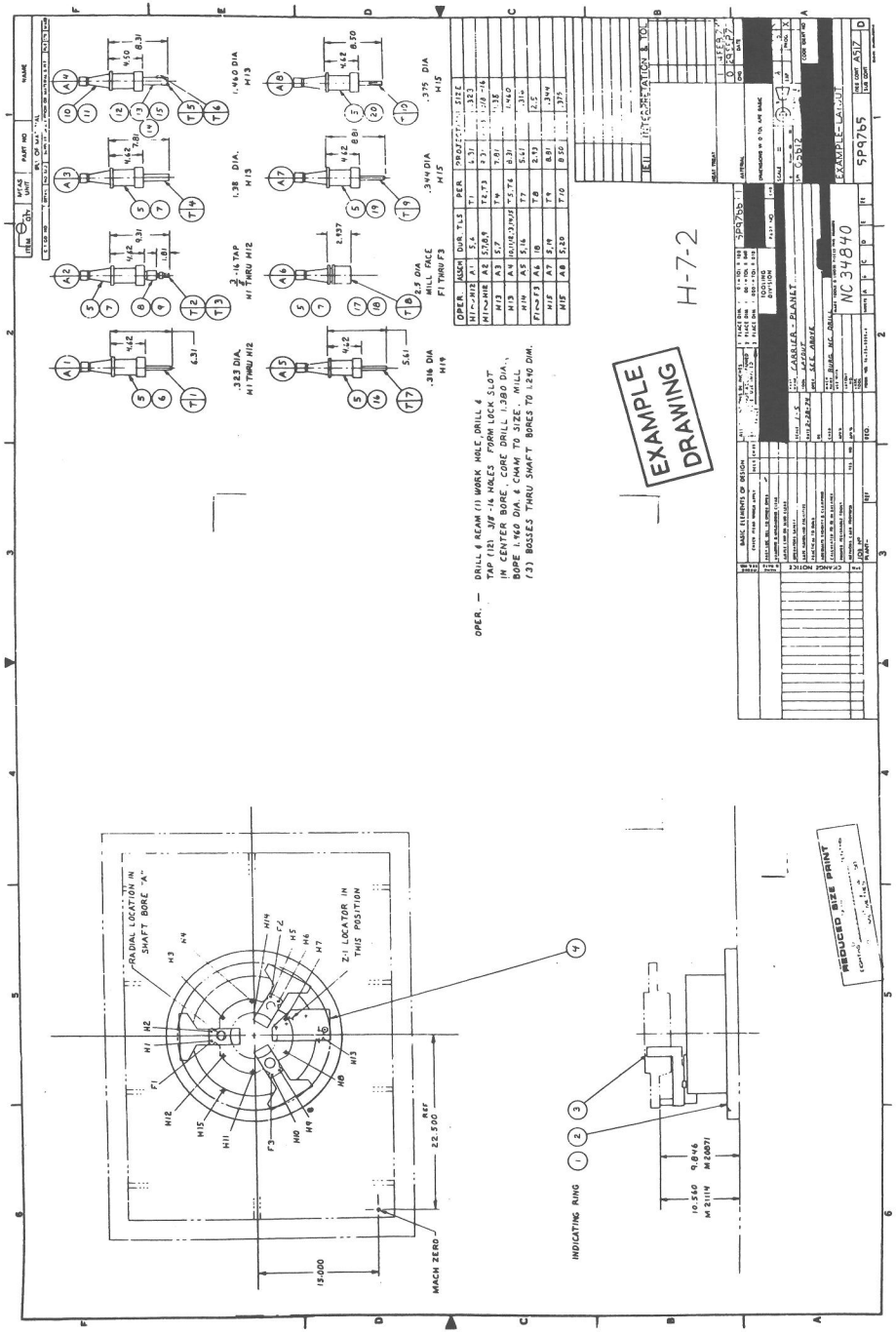


Figure 8.14 *Sample tool layout. Numerical Control Society, Glenview, IL*

to achieve this objective will vary, but the programmer should always remember that it is a very important aspect of his or her work.

PART PROGRAMMING PROCEDURE

The blocks of data entered in a part program are numbered N01, N02, N03, and so on. On completion of a machining program it is usually necessary to return to the beginning so that another component can be machined. The return to the program start position is usually achieved via a "rewind" or "return to start" command included at the end of the program.

With word address systems, this command is entered as a miscellaneous function designated M30, which has the effect of stopping all slide and spindle movement, turning off the coolant supply and rewinding the tape. When the tape has merely been used to transfer a program into the microcomputer memory, then it rewinds the program within the microcomputer. The stage at which this rewind must cease has to be identified, and this is achieved via a "rewind stop" program entry signified by the % sign. This is usually the first entry in a word address program.

With the start of the program established, the next three or four blocks of data will concern setting the machine controller so that it interprets subsequent data in the correct manner. These set-up entries include instructions relating to the following:

(a) *units,* which may be programmed in inch or metric;
(b) *slide movement,* which may be stated as incremental or absolute dimensional values;
(c) *speed,* which may be programmed as surface speed in feet/meters per minute or spindle speed in revolutions per minute;
(d) *feed,* which may be programmed as inches/millimeters per spindle revolution or inches/millimeters per minute.

Having established the basic set-up data, it may be helpful now to list in a general way the functions and machine movements necessary to produce the component. Consider the drawing for Exercise 1 (in Appendix C) and imagine that the machine is set with the spindle in its 'home' or 'base' datum position, that is, at a point some distance above the *XY* datum indicated on the drawing. Starting from this position, the part program must provide for the following:

1. Rapid linear movement to P1 in *X* and *Y*.
2. Rapid linear movement to a clearance position above *Z*0.

3. Spindle on clockwise direction.
4. Coolant on.
5. Feed linear movement to *Z* depth.
6. Rapid linear movement to clearance above *Z*0.
7. Rapid linear movement to P2.
8. Feed linear movement to *Z* depth.
9. Rapid linear movement to clearance above *Z*0.

. . . and so on.

These simple comments can, providing a space exists, be entered directly onto a program sheet or, if the program is being listed on plain paper, alongside each item of data, but it is probably a better plan to prepare a rough list in the first instance and then check carefully to ensure nothing has been overlooked. Relative codes and data can then be added to each statement.

Should it be found that, on completion of a program, omissions have inadvertently been made, the error can be rectified more easily if the block numbers are allocated in increments of five: N01, N05, N10, N15. It is then a simple matter to include additional blocks—N06, N07, N08, for instance—between N05 and N10.

If the program is being listed on a computer the blocks can be numbered consecutively, since any omission entered via the keyboard will automatically cause the existing blocks to renumber or, alternatively, renumbering can be easily effected. Many MDI control systems also have this facility.

A methodical approach to part programming is essential, and it is recommended that, even for a simple component, an operation schedule listing the tooling speeds and feeds to be used should be completed in the first instance.

WORD ADDRESS PROGRAMMING

Word address programming is largely based on an International Standards Organization (ISO) and Electronic Industries Association (EIA) code that require the program to be compiled using codes identified by letters, in particular G and M. Each code addresses, or directs, the item of data it precedes to perform a certain function within the control system.

The ISO and EIA Standards provided for 99 G codes and an identical number of M codes, each being exprssed by the address letter followed by two digits.

Not all the codes were allocated a specific function in the Standard and this gave the manufacturers of control systems the opportunity to introduce their own variations. There is, therefore, no standard word address machine programming language, although many of the recommendations made have been widely adopted.

The G codes, or preparatory functions, are used to set up the machine control unit modes of operation required for the machining that is to be carried out—whether movement is to be in a straight line/linear or radially/circular, for example. In general they relate to slide motion control. Examples of commonly used G codes are as follows:

G00 Rapid linear positioning, point to point
G01 Linear positioning at a controlled feed rate
G02 Circular interpolation, clockwise
G03 Circular interpolation, counter-clockwise
G04 Dwell for programmed duration
G33 Thread cutting, constant lead
G34 Thread cutting, increasing lead
G40 Cutter compensation, cancel
G41 Cutter compensation, left
G42 Cutter compensation, right
G70 Inch programming
G71 Metric programming
G80 Series associated with drilling, boring, tapping and reaming.

(For a complete list of G codes refer to Chapter 6.)

G codes may be "modal," that is, they remain active until cancelled. Alternatively they may be nonmodal, and are only operative for the block in which they are programmed.

The M codes, or miscellaneous functions, are used to establish requirements other than those related to slide movement. For example, they are used to activate spindle motion or to turn on a coolant supply. Examples of commonly used M codes are as follows:

M00 Program stop
M01 Optional stop
M02 End of program
M03 Spindle on clockwise
M04 Spindle on counter-clockwise
M05 Spindle off
M06 Tool change
M08 Coolant on
M09 Coolant off
M30 End of tape

(For a complete list of M functions refer to Chapter 6.)

As with G codes, some M functions are modal, remaining active until cancelled. M functions may also become active immediately upon reading of the

block or after all block commands are completed. (Refer to machinery manuals to determine how various codes operate.)

In addition to the address letters G and M there is also common usage of S, F, and T to indicate speeds, feeds, and tooling. The letter N is always used to identify block numbers.

The distinction between word address and conversational programming is best appreciated by reference to the simple movements discussed earlier.

To program the linear movement of −39.786 mm or −1.6 in. in the *X* axis using the word address technique, it is first necessary to establish the operating mode required. This is done by including the appropriate G code, in this case G01. Thus the complete program entry for the required move will be:

Inch N260 G01 X-1.6
Metric N260 G01 X-39.786

Similarly, reconsider the 0.3 in. or 8 mm radial movement through an arc of 90°. Once again the mode of operation has to be established using the appropriate G code, which for circular movement in a clockwise direction is G02. It will also be necessary to define the target position in the appropriate axes and also the start of the arc in relation to the arc center using I, J, and K address letters that correspond to the *X*, *Y*, and *Z* axes respectively. A word address progrm entry to achieve this movement would read as follows:

Inch N350 X1.7 Z-3.0 K.3
Metric N350 G02 X43.765 Z-75.000 K8

There are variations in procedure even when word address programming such a common machining feature as a radius. On some control systems the arc center may have to be defined—still using the I, J, and K address letters—in relation to the program datum and not the start position.

The programming of radial movements using the word address method will be returned to later in the text.

A word address program that includes a number of codes in both inch and metric is listed below. The program relates to the component detailed in Figure 8.15, and is typical of its type. The comments written alongside the data should convey to the reader an impression of how, prior to programming, the machining of a component is first broken down into operations. It also shows how the necessary machine control data are presented. Later in the text further reference will be made to the program to illustrate specific programming techniques and features.

PROGRAMMING EXAMPLE

Figure 8.15 shows a simple turned component for which a part program is to be prepared using the following basic programming information. (Examples of more detailed programming specifications are given in Appendix C.)

PREPARATORY FUNCTIONS (G CODES)

G00 Rapid movement
G01 Linear interpolation—movement at a programmed feed rate
G02 Circular interpolation, clockwise
G03 Circular interpolation, counter clockwise
G40 Cancel tool nose radius compensation
G41 Tool nose radius compensation left
G42 Tool nose radius compensation right
G70 Inch units
G71 Metric units

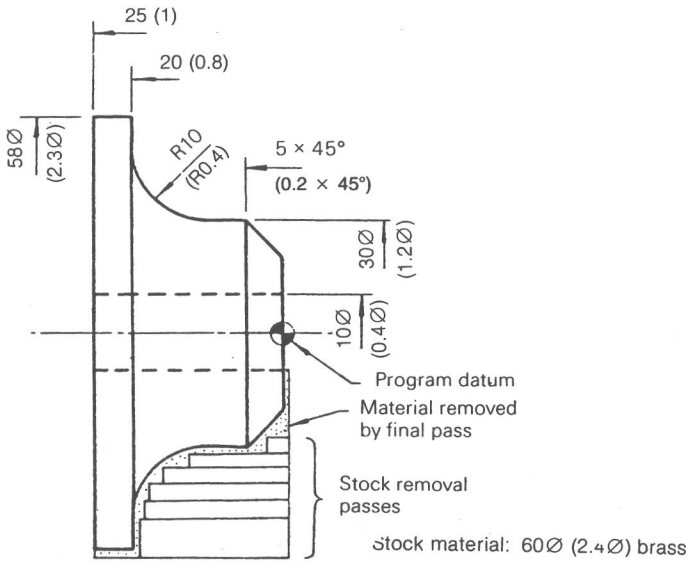


Figure 8.15 *Component detail. (Inch units are given in parentheses.)*

G90 Absolute positioning data } X axis values to be
G91 Incremental positioning data } programmed as diameters
G94 Feed (in. or mm)/min
G95 Feed (in. or mm)/rev
G96 Constant surface cutting speed
G97 spindle speed rev/min

MISCELLANEOUS FUNCTIONS (M CODES)

M00 Program stop
M01 Optional stop
M02 End program
M03 Spindle on clockwise
M04 Spindle on counterclockwise
M05 Spindle off
M06 Tool change
M08 Coolant on
M09 Coolant off
M30 End of program

OPERATION SCHEDULE

The first stage in the programming process is to prepare an operation schedule. An operation schedule for the component is shown in Figure 8.16, where only metric units are shown. The spindle speeds and feed rates have been determined by reference to the cutting data given in Chapter 7.

TOOLING INFORMATION

Although the component is a relatively simple one, it is still necessary to provide tooling information for the machine tool setter. This information is detailed on the form illustrated in Figure 8.17, where only metric units are shown.

PROGRAM LISTING

Attention can now be given to listing the necessary programming data, together with appropriate remarks to ensure a logical approach is being adopted and to

OPERATION SCHEDULE

PART No. Ex. 1 | DESCRIPTION PLUG | SHEET No. 1 OF 1
MACHINE TYPE HB370 | COMPILED BY A.R.C. | DATE 9-2-84

OP No.	DESCRIPTION	TOOLING TYPE AND SIZE	WORK HOLDING	CUTTING SPEED	FEED RATE	SPINDLE SPEED
1	CENTRE DRILL	HSS No 2 C/DRILL		28 M/MIN	.12 MM/REV	1500
2	DRILL	HSS DRILL Ø10	Ø60 COLLET	28 M/MIN	.18 MM/REV	890
3	TURN PROFILE	CEM. CARB. INSERT		170 M/MIN	.25 MM/REV	1350
4	PART OFF	CEM. CARB. INSERT		170 M/MIN	.16 MM/REV	1350

Figure 8.16 *(Metric units)*

TOOL PREPARATION AND SETTING DATA | PART No. EX. 1

TURRET POSITION	OFFSET No.	OPERATION	INSERT TYPE	HOLDER TYPE	PRE-SET LENGTHS X	PRE-SET LENGTHS Z
1	01	CENTRE DRILL	—	RC 107		
3	03	DRILL Ø 10	—	RC 110	DETERMINE	
4	04	TURN TO PROFILE	P10	TN 22-08	AND ENTER OFFSETS ON	
5	05	PART OFF	P20	GR 18-04	THE MACHINE	

Figure 8.17 *(Metric units)*

ensure that nothing is overlooked. The required program is listed below. (Note that, in this particular case, a programming form is not being used, but partially completed programming exercises involving the use of a form are given in Appendix C.)

PART PROGRAM (INCH)

	Data	Remarks
N10	G70 G90	Absolute inch
N15	G95 G97	Feed inches/rev Spindle speed rev/min
N20	G92 X4.0 Z8.0	Pre-set safe turret indexing position
N25	T0101 M06	Tool change. Tool No. 1. Off-set No. 1
N30	S3000 M03	Spindle on clockwise
N35	G00 X0 Z.1 M08	Rapid to start position. Coolant on
N40	G01 Z-.3 F.003	Center drill
N45	G00 Z.1	Rapid retract
N50	X4.0 Z8.0	Return to turret index position
N55	T0202 M06	Tool change. Tool No. 2 Off-set No. 2
N60	S2380 M03	Spindle Speed
N65	G00 X0 Z.1	Rapid to start position
N70	G01 Z-1.2 F.015 M08	Drill through .4ø
N75	G00 Z.1	Rapid retract
N80	X4.0 Z8.0	Return to turret index position
N85	T0303 M06	Tool change. Tool No. 3 Off-set No. 3
N90	S1430 M03	Spindle speed
N95	G00 X1.97 Z.1	Rapid to start position
N100	G01 Z-.768 F.020 M08	
N105	G00 X2.05 Z-.688	Rapid retract to clear cut surface
N110	Z.1	Second rough pass—start position
N115	X1.772	
N120	G01 Z-.748	
N125	G00 X1.85 Z-.669	
N130	Z.1	
N135	X1.575	Third rough pass—start position
N140	G01 Z-.709	
N145	G00 X1.654 Z-.63	
N150	Z.1	
N155	X1.417	Fourth rough pass—start position
N160	G01 Z-.65	
N165	G00 X1.496 Z-.57	
N170	Z.1	
N175	X1.26	Fifth rough pass—start position
N180	G01 Z-.512	
N185	G00 X1.339 Z-.433	
N190	Z.1	
N195	X.984	Sixth rough pass—start position
N200	G01 Z-.079	
N205	G00 X1.063 Z.1	

Data		Remarks	
N210	G41	Cutter radius compensations	
N215	X0	Rapid to *X* zero	
N220	G01 Z0	Rapid to *Z* zero	
N225	S1750 M03	Spindle speed and feed rate change	
N230	X1.0 F.007	Machine face to 1.0ø	
N235	X1.2 Z-.2	Machine chamfer	
N240	Z-.4	Linear move to radius start	
N245	G03 X2.0 Z.8 I.4	Circular interpolation	
N250	G01 X2.3	Linear move to 2.3ø	
N255	Z-1.1	Linear move to length	Finish machine profile
N260	G00 X2.5	Lift from finished surface	
N265	G40	Cancel cutter radius compensation	
N270	X4.0 Z8.0	Return to turret index position	
N275	T0404 M06	Tool change. Tool No. 4 Parting Tool	
N280	S1430 F.007	Spindle speed and feed rate change	
N285	G00 X2.5 Z-1	Rapid to start	
N290	G01 X.08 M08	Part off leaving stock faced	
N295	G00 X4.0 Z8.0	Return to turret index position	
N300	G92 X0 Z0 M30	Program end. Spindle and coolant off	

Note: To simplify the program, neither tool nose radius or thickness were used.

PART PROGRAM (METRIC)

Data		Remarks
N10	G71 G90	Absolute metric
N15	G95 G97	Feed mm/rev Spindle speed rev/min
N20	G92 X100 Z200	Pre-set safe turret indexing position
N25	T0101 M06	Tool change. Tool No. 1. Off-set No. 1
N30	S3000 M03	Spindle on clockwise.
N35	G00 X0 Z2 M08	Rapid to start position, coolant on
N40	G01 Z-8 F.1	Center drill
N45	G00 Z2	Rapid retract
N50	X100 Z200	Return to turret index position
N55	T0202 M06	Tool change. Tool No. 2 Off-set No. 2
N60	S2380 M03	Spindle speed
N65	G00 X0 Z2	Rapid to start position
N70	G01 Z-30 F.18 M08	Drill through 10ø
N75	G00 Z2	Rapid retract
N80	X100 Z200	Return to turret index position
N85	T0303 M06	Tool change. Tool No. 3 Off-set No. 3
N90	S1430 M03	Spindle speed
N95	G00 X50 Z2	Rapid to start position
N100	G01 Z-19.5 F.3 M08	
N105	G00 X52 Z-17.5	Rapid retract to clear cut surface
N110	Z2	
N115	X45	Second rough pass—start position

Data		Remarks	
N120	G01 Z-19		
N125	G00 X47 Z-17		
N130	Z2		
N135	X40	Third rough pass—start position	
N140	G01 Z-18		
N145	G00 X42 Z-16		
N150	Z2		
N155	X36	Fourth rough pass—start position	
N160	G01 Z-16.5		
N165	G00 X38 Z-14.5		
N170	Z2		
N175	X32	Fifth rough pass—start position	
N180	G01 Z-13		
N185	G00 X34 Z-11		
N190	Z2		
N195	X25	Sixth rough pass—start position	
N200	G01 Z-2		
N205	G00 X27 Z2		
N210	G41	Cutter radius compensations	
N215	X0	Rapid to *X* zero	
N220	G01 Z0	Rapid to *Z* zero	
N225	S1750 M03	Spindle speed	
N230	X20 F.15	Machine face to 20ø	
N235	X30 Z-5	Machine chamfer	
N240	Z-10	Linear move to radius start	
N245	G03 X50 Z-20 I10	Circular interpolation	
N250	G01 X58	Linear move to 58ø	
N255	Z-26	Linear move to length	Finish machine profile
N260	G00 X64	Lift from finished surface	
N265	G40	Cancel cutter radius compensation	
N270	X100 Z200	Return to turret index position	
N275	T0404 M06	Tool change. Tool No. 4 parting tool	
N280	S1430 F.18	Spindle speed and feed rate change	
N285	G00 X64 Z-25	Rapid to start	
N290	G01 X-2	Part off leaving stock faced	
N295	G00 X100 Z200	Return to turret index position	
N300	G92 X0 Z0 M30	Program end. Spindle and coolant off	

Note: To simplify the program no tool nose radius or thickness was used.

DATA FORMAT

Data are written in blocks. The data within a block were once expressed in a fixed sequence with each block containing all data (even if they have not changed from the previous block), but now almost exclusively the commands appear in random order without the repetition of unchanged data but with each word being clearly identified by its address letter. The terminology used to describe these two methods is "fixed block" and "variable block word address," respectively. The following examples illustrate these formats.

Fixed Block Example

N	G	X	Y	Z	I	J	K	F	S	M	Remarks
0250	00	05000	09000	04000	00000	00000	00000	2000	0350	03	Rapid position *X*, *Y*, *Z*, and start spindle.
0300	01	08500	09000	04000	00000	00000	00000	0310	0350	08	Mill in *X* axis—turn coolant on
0350	00	08500	09000	04500	00000	00000	00000	2000	0350	09	Retract *Z* axis—turn coolant off

Note: When information is placed in machine format, no spaces occur between data words.

Variable Block Example (spaced out in a form)

N	G	X	Y	Z	I	J	K	F	S	M	Remarks
N0250	G00	X05000	Y09000	Z04000	—	—	—	F2000	S0350	M03	Rapid position *X*, *Y*, *Z*, and start spindle
N0300	G01	X08500	—	—	—	—	—	F0310		M08	Mill in *X* axis—turn coolant on
N0350	G00	—	—	Z04500	—	—	—	F2000		M09	Retract *Z* axis—turn coolant off

Variable Block Example (without form—using decimal point format)

N0250 G00 X5 Y9 Z4 F200 S350 M03	Rapid position *X*, *Y*, *Z*, and start spindle.
N0300 G01 X8.5 F31 M08	Mill in *X* axis—turn coolant on.
N0350 G00 Z4.5 F200 M09	Retract *Z* axis—turn coolant off.

It is necessary for the part programmer to be aware of the data format for the system being used, and also to be familiar with the classificaton of the data that dictates the way in which it may be presented within a block. For example, a programming manual could indicate that data must conform to the following classification:

N4, G2, X3/3, Y3/3, Z3/3, F4, S4, T2, M2 (METRIC)
N4, G2, X2/4, Y2/4, Z2/4, F4, S4, T2, M2 (INCH)

This classification indicates the following:

N4	The block sequence address letter N may be followed by up to four digits.
G2	The preparatory function address letter G may be followed by up to two digits.
X3/3, Y3/3, Z3/3 (Metric) X2/4, Y2/4, Z2/4 (Inch)	The axis identification letters X, Y, and Z may be followed by up to three digits in front of the decimal point, and up to three after in metric form. Identification letter may be followed by two digits in front of decimal point, and up to four after in inch format. (Dimensional values may be subject to other limitations as explained below.)
F4	The feed address letter F may be followed by up to four digits.
S4	The spindle speed or cutting speed address letter S may be followed by up to four digits.
T2	The tool address letter T may be followed by up to two digits.
M2	The miscellaneous function address letter M may be followed by up to two digits.

(Note: These formats will change from one machine control to another. For a complete list of your machine code formats refer to your machine tool manual.)

The description above has stated that up to so many digits may be used. Some systems require that leading zeros are included and some do not. Thus a linear slide movement at a programmed feed rate may be programmed as G01 or G1, depending on the system used.

Similarly, dimensional values may also have to be programmed according to certain rules. For instance, using a data classification of 3/3 it would be possible, depending on the requirements of the system, to program a value of 32 mm in a number of ways:

(a) 032000—all digits must be included but no decimal point.
(b) 32000—leading zeros are omitted, but no decimal point is required; trailing zeros must be included.
(c) 32.000—the decimal point and all trailing zeros are required.
(d) 32.—no leading or trailing zeros are required but the decimal point must be included.

(e) 32—whole numbers may be programmed without leading or trailing zeros and without a decimal point.

SLIDE MOVEMENTS

Both word address and conversational programming require definiton of the slide movements necessary to position the cutting tool correctly in relation to the work.

This positioning is described in three ways:

(a) point-to-point;
(b) linear interpolation;
(c) contouring/circular interpolation.

Point-to-point positioning involves programming instructions that identify only the next relative tool position required. The position may be reached by movement in one or more axes at a rate of travel that is generally, though not necessarily, the maximum for the machine. If metal cutting takes place during this type of motion, it must be in one axis or the cut path will not necessarily repeat cycle to cycle.

Figure 8.18 shows details of a component. To drill the holes in this component would require two-axis point-to-point positioning. Note that the positioning prior to drilling is clear of obstruction, therefore path is not important and the actual drilling is a single-axis move, so point to point can be used. Note that point-to-point positioning is not capable of machining angles or contours, because they require controlled movement of more that one axis.

Linear interpolation control requires programmed instructions that specify

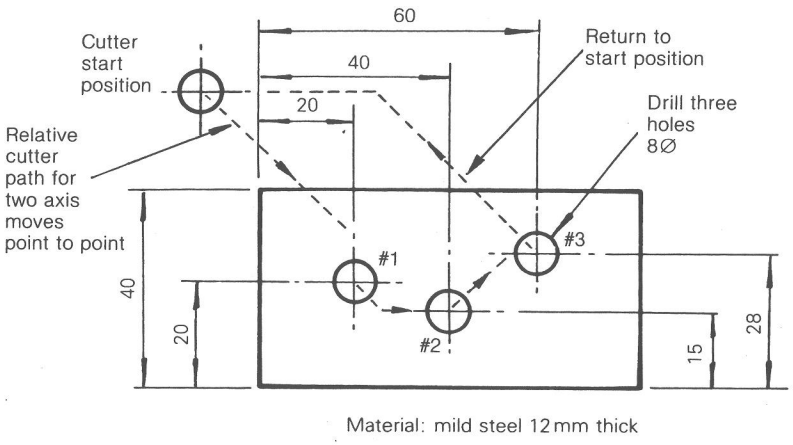


Figure 8.18 *Component detail requiring point-to-point positioning to drill holes.*

both the next position and the rate of travel, or feed rate, to be employed to reach that position; the resulting cutter path is a straight line. Metal cutting would normally take place during such a move. Linear interpolation allows the machining of straight lines at a feed rate using one or two axes of motion. Other more expensive machines allow linear interpolation in three axes simultaneously. Figure 8.19 illustrates examples of one- and two-axis linear interpolation moves.

Contouring is used to describe movements involving at least two slides. The movements occur simultaneously and at a predetermined feed rate, and result in a continuous machining path which is not a straight line. An elliptical profile or a combination of arcs—the production of an arc being referred to as 'circular interpolation'—are good examples of contouring. Contouring will normally refer to irregular curves that must be machined using minute straight line segments to generate it. Contouring requires many data blocks and multiple axis movement capability. Circular interpolation, on the other hand, will produce uniform arc segments or circles with minimal programming owing to the machine control's ability to self-generate uniform arc data. The principle is illustrated in Figure 8.20.

DEFINITION OF THE AXES OF MOVEMENT

Whether conversational or word address programming is being used, the direction in which slide movement is to occur is defined by a letter, which for common machines is either X, Y, or Z for linear movement and B for rotary table movement where applicable. Axes definition codes appear together with

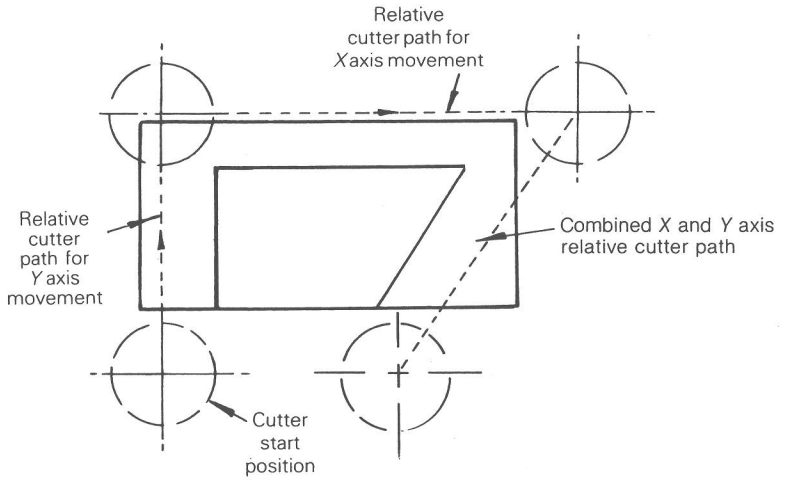


Figure 8.19 *Linear interpolation.*

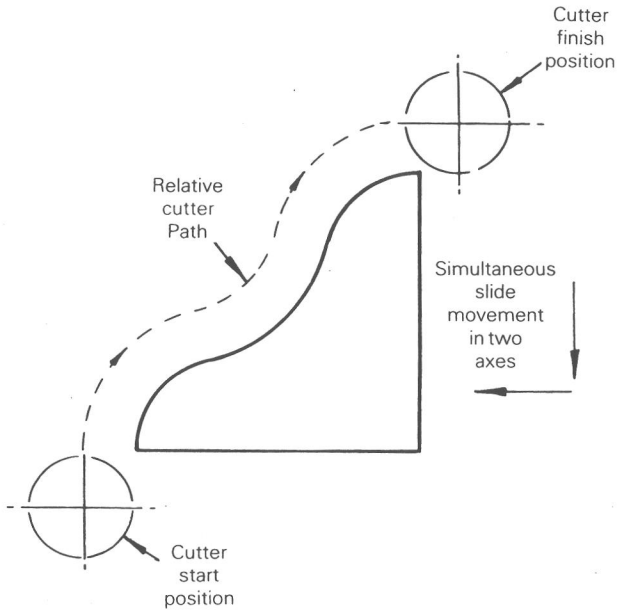


Figure 8.20 *Contouring.*

a positive (+) or negative (−) sign for determining direction. In practice the + sign is not actually entered, because if a sign is omitted, the control automatically assumes plus.

The definition of the axes of movement on common machine types, namely a turning center, a vertical machining center, and a horizontal machining center, are illustrated in Figure 8.21. Two points should be noted in relation to the illustrations. First, on a turning center having a rear-mounted tool post the plus (+) and minus (−) in the X axis would be reversed. Movement of the tool away from the spindle axis is always plus.

Second, the axes definitions shown indicate the *machine* slide movements. In the case of a turning center these movements are identical to the tool movement in relation to the work. On milling machines, where it is the table and not the cutting tool which moves, this is not the case. For programming purposes, where it is easier to imagine that the tool is moving, it is necessary to redefine some movements. On a vertical machining center, for example, in order to achieve a tool movement in relation to the workpiece in the X positive or plus direction it is necessary to build a machine slide that movement in the X axis would cause physical table movement to the right under the machine spindle viewed from the front of the machine. In determining axes directions, the programmer can always consider that the part is viewed through the tool from the shank to the tip. Right-theoretical tool motion on the part is always

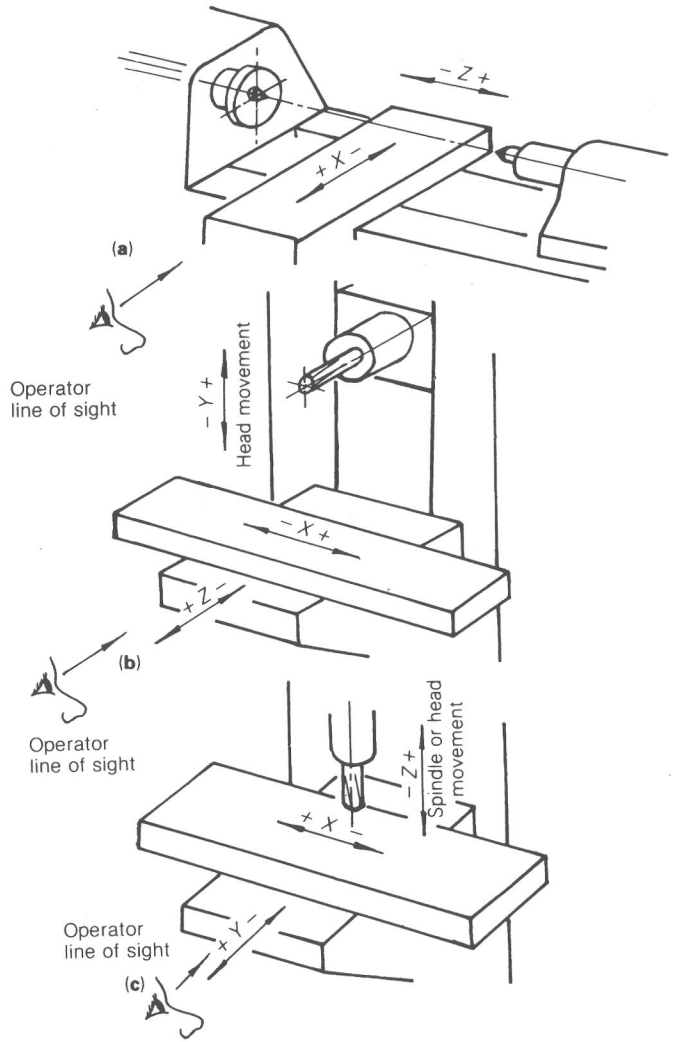


Figure 8.21 *Identification of slides and direction of the slide movement on common machine tools:* (a) *center lathe (turning center),* (b) *horizontal milling machine (horizontal machining center),* (c) *vertical milling machine (vertical machining center).*

positive, with left being negative. Tool motion viewed this way in the up direction on the part print is positive, with down being negative. Tool motion causing penetration of the tool into the work is negative, with retraction being positive. If rotary tables are involved, clockwise direction viewed looking into the table face from outside is positive, with counterclockwise being negative.

Note: these directions are based on a normal operator position in front of the machine.

In addition to linear movement, the production of a part may also require rotary movement which is provided by the use of ancillary equipment such as rotary tables and indexers. These movements are also controlled via the machining program and are identified by the letters A, B, and C as illustrated in Figure 8.22.

DATUMS

There are two datums involved in CNC machining that concern the part programmer.

The first of these is the program datum, which is established by the programmer when writing the program. This datum is at the intersection of the *X*, *Y*, and *Z* axes when milling, and at the intersection of the *X* and *Z* axes when turning. In both cases it is given the numerical identity of zero. The actual position of this datum in relation to the workpiece is optional, although there are certain factors to be taken into consideration.

The program datum is, in effect, the point from which the slide movements in each axis will be dimensionally related. It should be noted that part setup on a machine tool must allow for the proper machine zero/program zero relationship. When program data are stated in absolute terms (see below), all subsequent moves will also be dimensionally related to that point.

The second datum that concerns the part programmer is the machine datum. This datum is a set position for the machine slides where the axes intersect, and it has a numerical identity of zero within the control system.

On some machines the machine datum is a permanently established position (referred to as a fixed zero) and cannot be altered, although it can be repositioned on a temporary basis via a zero "shift" or "offset" facility. On other machines a new datum can be established anywhere within the operating pocket of the machine, a facility referred to as a "floating zero."

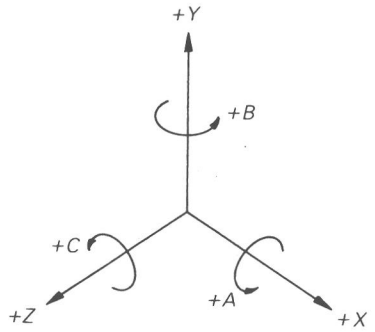


Figure 8.22 *Identification of rotary movements.*

Clearly, there must be some correlation between the machine datum and the program datum when setting the machine if the programmed slide movements are to achieve the intended effect, and the practicalities involved are discussed in more detail in Chapter 6.

ABSOLUTE AND INCREMENTAL POSITIONAL DATA

Once the direction of movement has been established, the distance moved by the machine slide to bring it to a desired position has to be defined dimensionally. This is achieved by the use of linear coordinates, with the dimensions being stated in absolute or incremental terms.

A third method is sometimes used, involving the use of polar coordinates. This requires a distance (the radius) stated in relation to a defined point and at an angle stated in relation to a datum axis. It generally requires the control system to include a special programming facility.

Absolute dimensional definition requires all slide movements to be related to a predetermined zero datum.

Incremental dimensional definition requires each slide movement to be related to the final position of the previous move.

Figure 8.23(a) shows the details of a turned component. The intersection of the spindle centerline and the face of the work is the program zero datum. Assume that a final trace of the component profile is to be programmed.

The dimensional definition in absolute and incremental values that would be required to define slide movement is shown in Figures 8.23(b) and 8.23(c) respectively.

In Figure 8.24(a) the details of a milled component are given. Absolute and incremental dimensional values required to program the machining of the slot are shown in Figures 8.24(b) and 8.24(c) respectively.

Earlier programming languages required dimensional data to be stated in *either* absolute or incremental terms. Modern controllers often provide a "mix and match" facility that permits the use of both within the same program, and even within the same data block. The distinction is achieved by the continued use of the G90 (absolute) and G91 (incremental) preparatory function codes, or X, Y, and Z word addresses for absolute values and U, V and W for incremental values. The use of the "G" code for changing from absolute to incremental is much more common.

CIRCULAR INTERPOLATION

Circular interpolation allows for programming a machine to move the cutting tool in a circular path with a uniform arc, with only a few commands. Circularlike motion is achieved through the machine controls capability to cal-

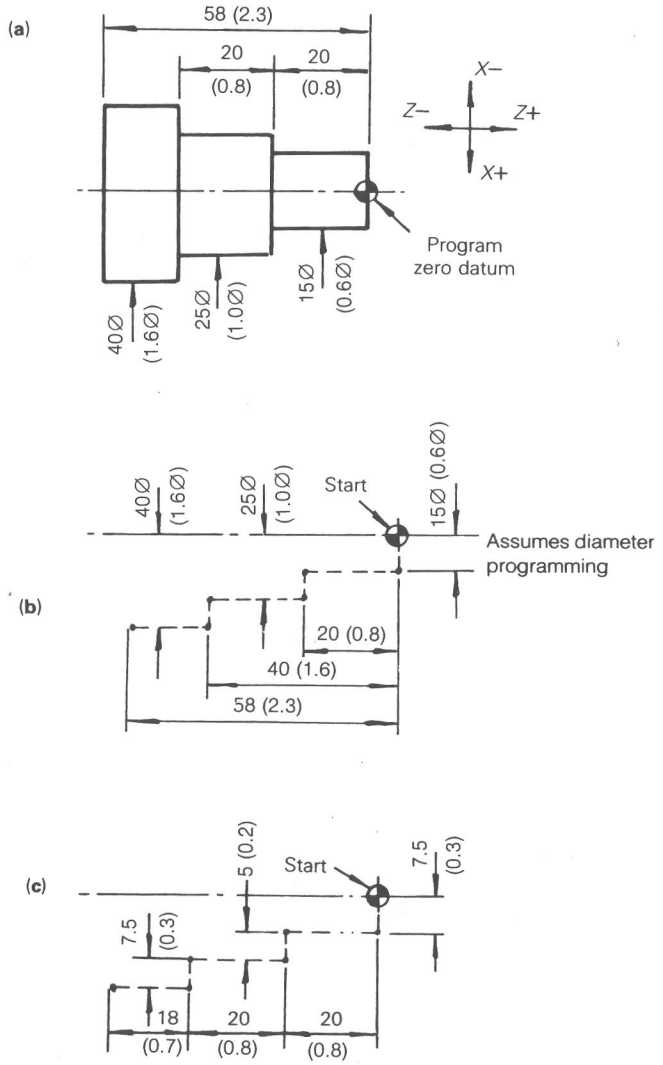


Figure 8.23 **(a)** *Component detail,* **(b)** *absolute dimensions, and* **(c)** *incremental dimensions. (Inch units are given in parentheses.)*

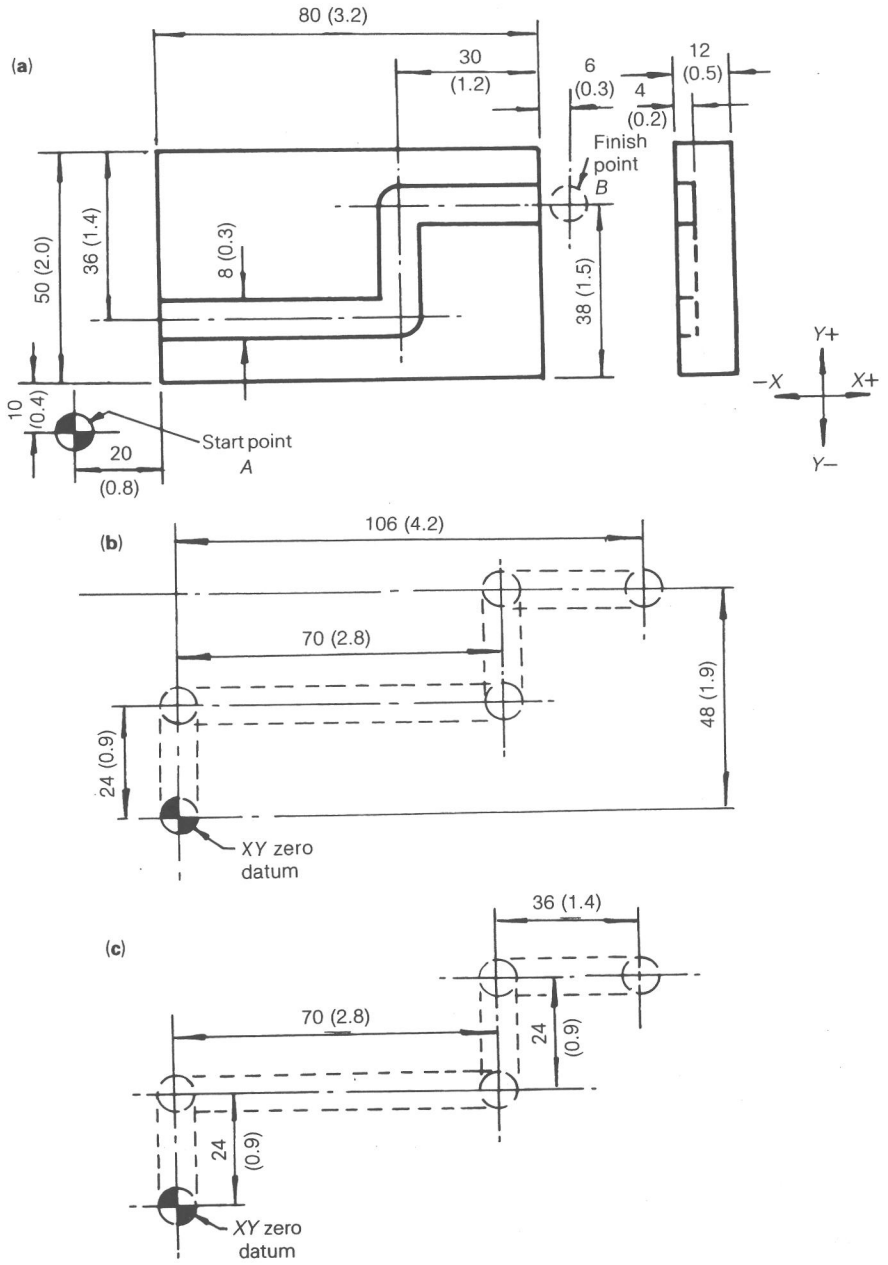


Figure 8.24 **(a)** *Component detail,* **(b)** *absolute dimensions, and* **(c)** *incremental dimensions. (Inch units are given in parentheses.)*

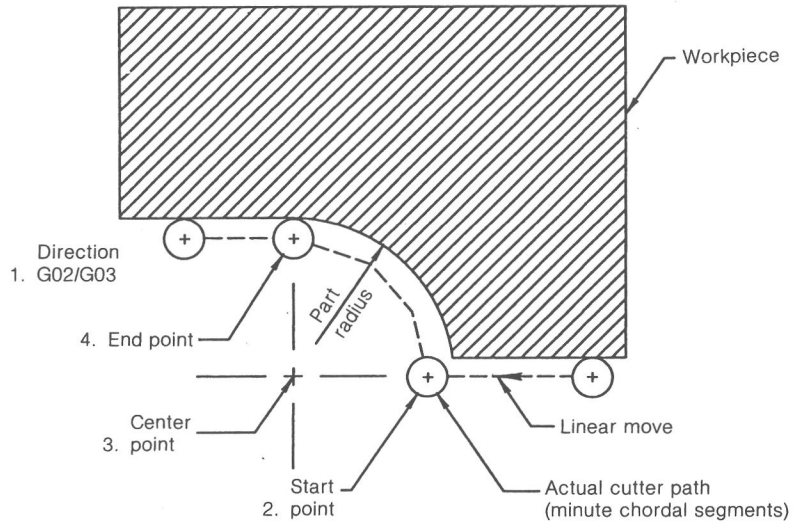


Figure 8.25 *The four basic elements of circular interpolation.*

culate minute cordal movements between commanded points (Figure 8.25). The basic elements of circular interpolation will deal with the following four items:

1. *Direction of rotation*—The direction of rotation that the cutter path will follow around the circle or arc segment.
2. *Start point*—The actual beginning point of the circle or arc segment where the cutter has previously been positioned.
3. *Center point*—The point of rotation in which the circle or arc segment is developed around.
4. *End point*—The finish point of the cutter path used to generate the circle or arc segment.

It was explained earlier in the text that circular arc programming on conversational MDI systems has been reduced to a simple data entry that specifies the target position, the value of the radius, and the direction of rotation. This simple method of defining circular movement in a single block with the direction of rotation being defined by the appropriate G code is also available on some word address systems (see method 3), but many systems employ one of two slightly more complex techniques of quadrant circular interpolation (methods 1 and 2).

Common to all programming systems is the need to determine whether the relative tool travel to produce a particular arc is in a clockwise (CW) "G02" or counterclockwise (CCW) "G03" direction, and the location of the arc or circle center. The following approaches are usually helpful:

1. For turning operations look down onto the top face of the cutting tool. (For inverted tooling this involves looking up at the tool from below.)
2. For milling operations look along the machine spindle centerline toward the surface being machined.

It should be noted that this technique does not always correspond with the definition adopted by the control system's manufacturer. A simple trial program entered into the machine will clarify the situation if it cannot be determined from the machinery manuals.

Determining clockwise and counterclockwise direction on milling machines with multiple plane circular interpolation capability can be done using the following rule and Figure 8.26.

Clockwise and counterclockwise are determined by looking at the plane of the circle from the positive end of the coordinate axis normal to the plane of the circle.

The arc center definition for circular interpolation is also a standard requirement. It informs the control of the point of rotation for an arc along standard axes from the arc start point or program datum. Use the following definitions and Figure 8.27 to determine proper arc center addresses needed for various planes of rotation.

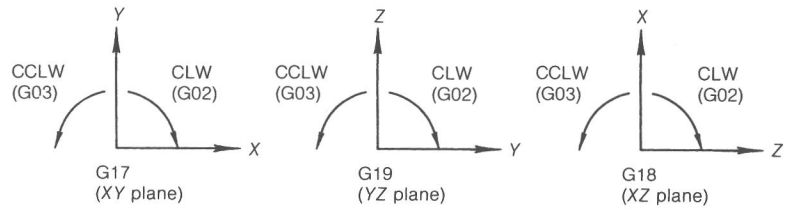


Figure 8.26 *Determining clockwise and counterclockwise directions.*

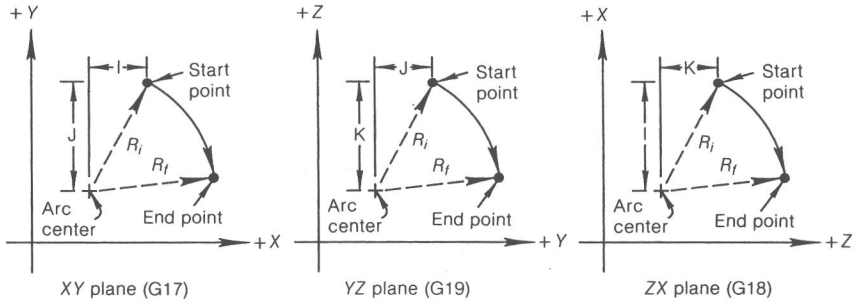


Figure 8.27 *Arc centers.*

Use of I, J, and K to Specify the Arc Center

I is the *increment* along the *X* axis from the start point of the arc to the arc center.
J is the *increment* along the *Y* axis from the start point of the arc to the arc center.
K is the *increment* along the *Z* axis from the start point of the arc to the arc center.

Note: Refer to your machinery manual to determine whether or not positive or negative signs will be required on the I, J, K values.

QUADRANT CIRCULAR INTERPOLATION

Quadrant circular interpolation allows up to 90° of arc to be programmed as long as the entire segment does not cross quadrant lines; refer to Figure 8.28

For any circle there are four quadrants that are created by axis lines which cross at the center of the circle. These axis lines are parallel to the part coordinate axes. No circular block may cross either of these axes. Starting at any point on the circle, it can be seen that the maximum number of circular controller data blocks to cut a 360° arc is five.

An individual circular controller data block cannot cross an axis line. To continue into the next quadrant, terminate the present circular controller data block and program another circular controller data block, starting on the axis. This procedure is quite straightforward.

A minimum of four sets of dimensional data are required to complete a full circle with up to four dimension words per block. Two dimension words are required to define the arc and one or two are required to define the arc center. Arc center values of zero are normally not required. Machines that are capable

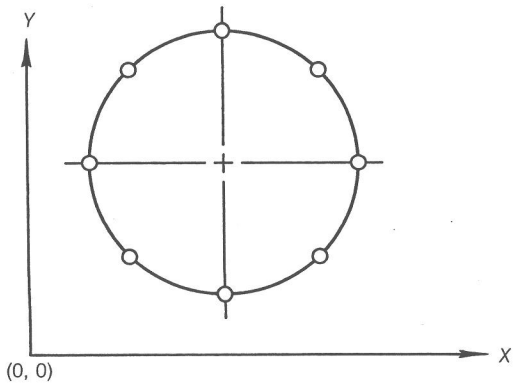


Figure 8.28 *Circle quadrants.*

of either quadrant or 360° circular interpolation will require a modal G code to be programmed (G74 quadrant; G75 360°). *Note:* For exact programming format and arc size limits refer to your machinery manual.

The two arc center address programming techniques previously referred to can be described using the component detail shown in Figure 8.29, and assuming that the last programmed move has brought the cutting tool to the start point indicated.

Method 1

1. The target or finish point of the arc is dimensionally defined, using *X*, *Y*, or *Z* values, in relation to the program datum when using absolute mode, or to the finish position from the previous move when using incremental mode.
2. The center of the arc is dimensionally defined in relation to the start point using I, J, and K values measured along the *X*, *Y*, and *Z* axes respectively.

Using this method the arc shown in Figure 8.29 would be programmed as follows:

Inch:
In absolute terms: G02 X1.6 Z1.6 I0 K.8 (Diameter programming of lathe part.)
In incremental terms: G02 X.8 Z-.8 I0 K.8

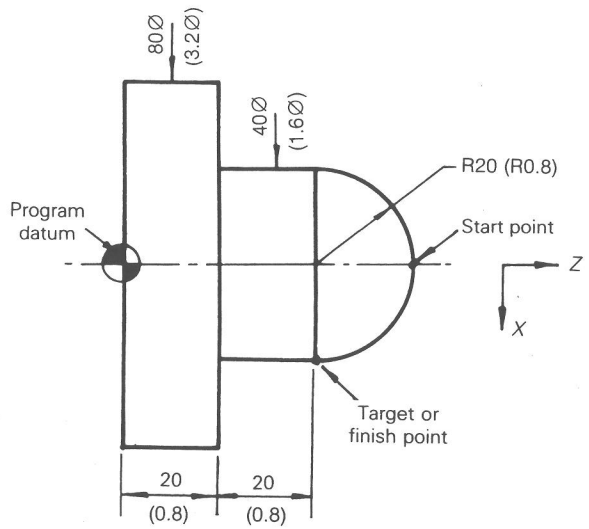


Figure 8.29 *Component detail. (Inch units are given in parentheses.)*

Metric:
In absolute terms: G02 X40 Z40 I0 K20 (Diameter programming of lathe part.)
In incremental terms: G02 X20 Z-20 I0 K20

Note that the I code has no value because the center and start point of the arc are in line with each other. In practice, when a value is zero, it is not entered into the program. In this method of arc definition the I, J, and K values are unsigned.

Method 2

The second method of word address arc programming differs in the way the arc center is defined. The following data are required:

1. The target or finish point of the arc is dimensionally defined, using X, Y, and Z values, in relation to the program datum when using absolute mode, or to the finish position from the previous move when using incremental mode.
2. The center of the arc is dimensionally defined in relation to the program datum using I, J, and K values measured along the corresponding X, Y, and Z axes, respectively, in absolute mode. The arc center is still defined from the previously defined point in incremental mode.

Using this second method of programming the arc shown in Figure 8.30 would be programmed as follows:

Inch:
In absolute terms: G02 X1 Y2.4 I1 J1.4
In incremental terms: G02 X1 Y1 I1 J0

Metric:
In absolute terms: G02 X25 Y60 I25 J35
In incremental terms: G02 X25 Y25 I25 J0

It is possible when using this approach for the I, J, and K values to be negative, as illustrated in Figure 8.31. These values are, therefore, signed plus or minus.

The two arc programming methods described will cater for movement within one quadrant only with each block of program data. Thus programming a complete circular move would require four blocks of data minimum. Similarly, blending arcs would require a separate block of data for each quadrant involved. This latter situation is illustrated in Figure 8.32; the first block would take the tool from point A to point B, and the second block would continue the movement from point B to point C.

When the start or stop points or both do not coincide with an *X*, *Y*, or *Z* axis, that is the arc is not exactly 90° or a multiple of 90°, it will be necessary to make a series of calculations.

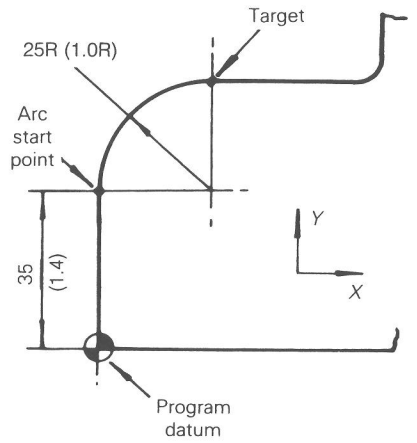


Figure 8.30 *Milled component detail. (Inch units are given in parentheses.)*

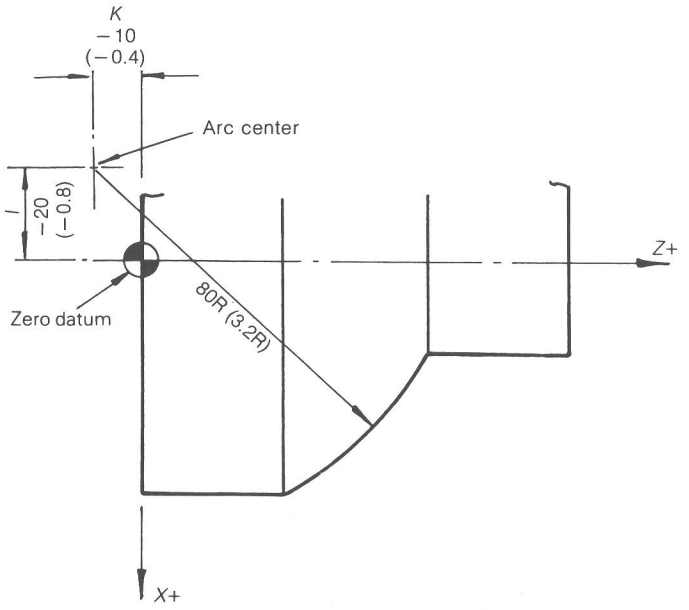


Figure 8.31 *Circular interpolation: negative I and K values for an absolute program. (Inch units are given in parentheses.)*

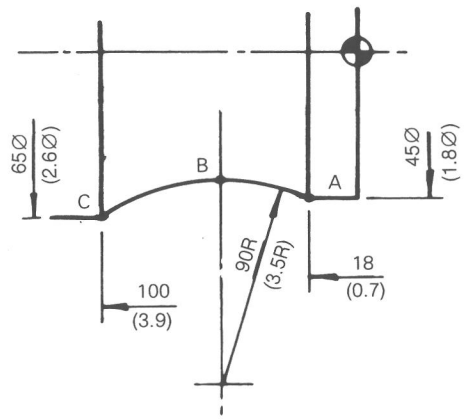


Figure 8.32 *Profile detail requiring two arcs to be programmed. (Inch units are given in parentheses.)*

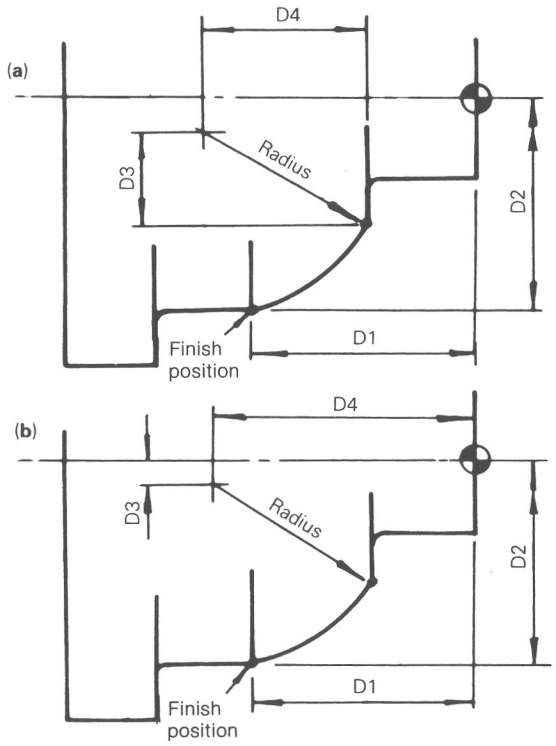


Figure 8.33 *Calculations for arcs less than 90°: (a) method 1; (b) method 2. (Inch units are given in parentheses.)*

Consider the part detail shown in Figure 8.33(a) and again in Figure 8.33(b), each diagram relating to an arc programming method as indicated. Whichever of the two methods is used, the finish arc position indicated by the dimensions D1 and D2 will have to be defined dimensionally. It will also be necessary to calculate the additional dimensions indicated on each drawing as D3 and D4, these latter dimensions being expressed in the part program as I and K values. A number of circular interpolation examples follow.

PROGRAMMING EXAMPLES

The circle in Figure 8.34 will be used to illustrate programming various arcs, clockwise and counterclockwise, absolute and incremental. (All units in the following examples are in inches.)

Example 1 (Method 1)

Go from P2 to P5 clockwise (absolute) (G74 default).

Signed I and J	Unsigned I and J
G17	G17
G90	G90
G2X3Y2I-.7071J-.7071F10	G2X3Y2I.7071J.7071F10
X2Y1I-1	X2Y1I1
X1Y2J1	X1Y2J1

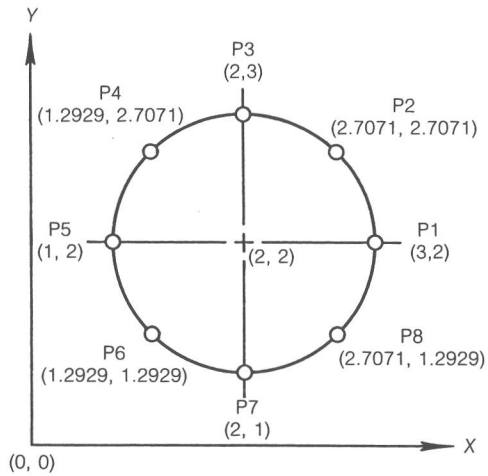


Figure 8.34 *Example of programming arcs.*

Example 2 (Figure 8.35)

Go from P2 to P5 counterclockwise (incremental) (G74 default).

Signed I and J	Unsigned I and J
G17	G17
G91	G91
G3X-.7071Y.2929I-.7071J-.7071F10	G3X-.7071Y.2929I.7071J.7071F10
X-1Y-1J-1	X-1Y-1J1

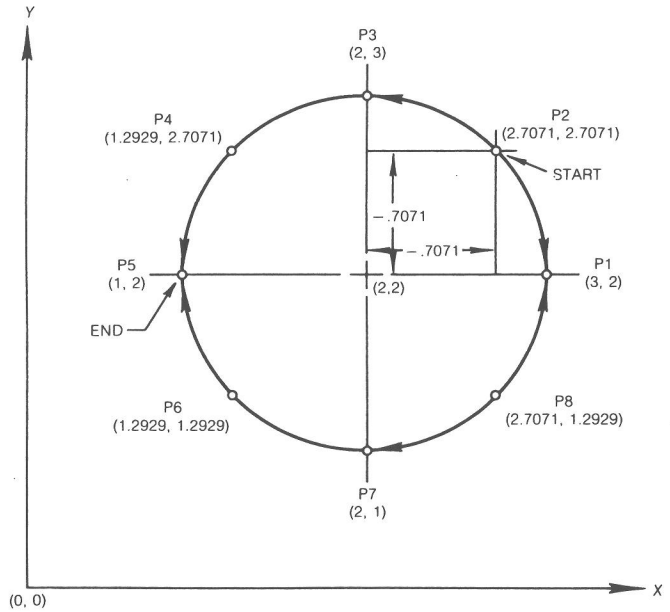


Figure 8.35 *Examples 1 and 2.*

Example 3 (Figure 8.36)

Go from P4 to P7 clockwise (incremental) (G74 default).

Signed I and J	Unsigned I and J
G17	G17
G91	G91
G2X.7071Y.2929I.7071J-.7071F10	G2X.7071Y.2929I.7071J.7071F10
X1Y-1J-1	X1Y-1J1
X-1Y-1I-1	X-1Y-1I1

Example 4 (Method 1) (Figure 8.36)

Go from P4 to P7 counterclockwise (absolute) (G74 default).

Signed I and J	Unsigned I and J
G17	G17
G90	G90
G3X1Y21I.7071J-.7071F10	G3X1Y2I.7071J.7071F10
X2Y1I1	X2Y1I1

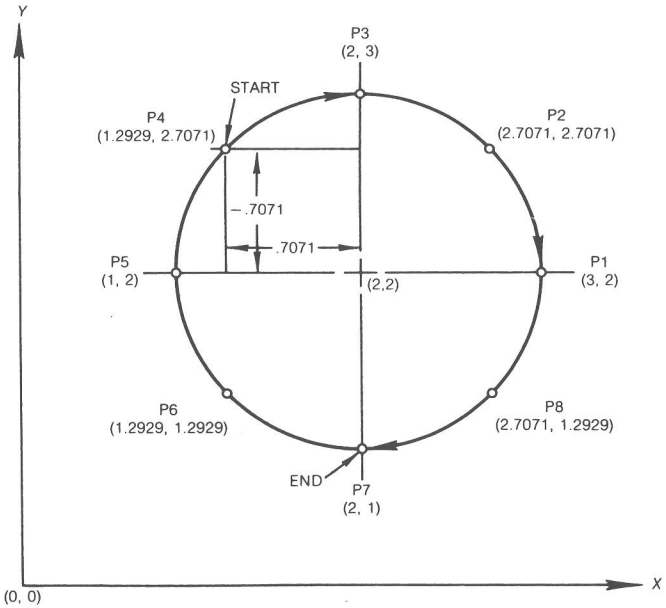


Figure 8.36 *Examples 3 and 4.*

Method 3

Single block or 360° circular interpolation is now available on more sophisticated CNC controls. Single block circular interpolation allows up to 360° of circular movement to be programmed in a single block of information when the modal G75 code is active. All previous circular interpolation command words defined are still used to determine rotation, finish/end position, and arc

center. It should be noted, though, that in this form of circular interpolation the I, J, and K combination used must be signed to determine the circle's center.

Another difference that will be noted in this method of programming is if a full 360° of circle is to be produced, the starting and ending points of rotation will be identical. When identical starting and ending points occur, the end point of the arc does not have to be programmed, only the rotation and arc center information are required; refer to the examples with Figure 8.37.

Example 5 (Method 3) (Figure 8.37)

Go from P3 to P3 clockwise.

Incremental	Absolute
G17	G17
G91	G90
G75	G75
G2I0J-1F10	G02I0J-1F10

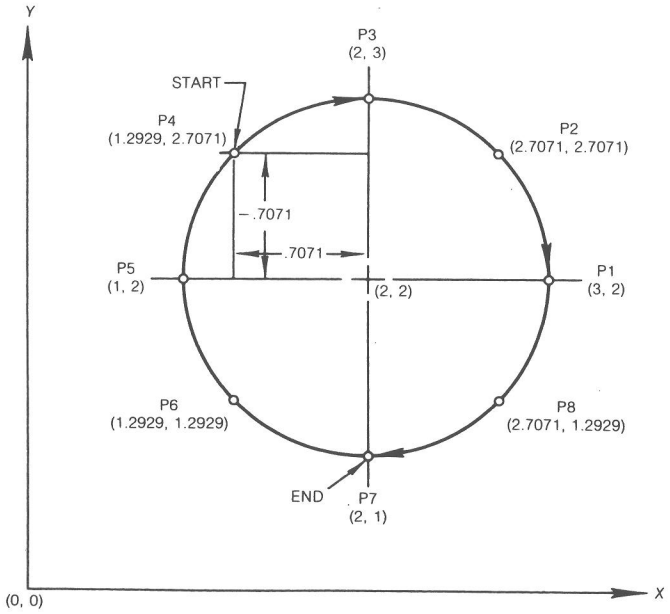


Figure 8.37 *Examples 5 and 6*

Example 6 (Method 3) (Figure 8.37)

Go from P4 to P7 clockwise.

Incremental	Absolute
G17	G17
G91	G90
G75	G75
G2X.7071Y-1.7071I.7071J-.7071F10	G2X2Y1I.7071J-.7071F10

TOLERANCED DIMENSIONS

It is often the case that dimensions on drawings are toleranced, thus permitting a higher and lower limit. Since it is only possible to enter one value into the control unit, it is logical that this should be the middle or mean value of the tolerance band.

REPETITIVE MACHINING SEQUENCES

There are a number of repetitive sequences that are commonly used when machining a variety of components like drilling a hole. Other less common sequences are also repetitive, but on only one particular component (hole patterns—milling path). It is helpful, since it reduces the program length and also simplifies programming, if such a sequence can be programmed just once; it is then given an identity so that it can be recalled into the program as and when required.

Repetitive machining sequences can be generally classified as follows:

(a) Canned or fixed cycles that are a built in feature of the machine control system.
(b) User or programmer defined routines to suit the particular job in hand (custom canned cycles, program macros).

The facility for the programmer to devise special routines may be restricted, especially on small training machines. However, even the most simple system will usually include one or two canned cycles. The controls fitted to advanced machines will have many available.

CANNED CYCLES

In a text of this nature it would be impossible to deal in detail with all the canned cycles that are available, but the review that follows will convey a good impression of the range currently in use and likely to be encountered. More

specifically, the student is advised to make a careful study of the programming methods and techniques associated with the control system he or she will be using. In this respect a close examination of the examples found in the programming manuals will be found to be helpful. The point to remember is that the use of canned cycles is an aid to programming efficiency and accuracy, and they should be used whenever possible. You will now notice that the previously mentioned special cycles for conversational programming are not necessarily unique to that form of programming. Word address programming has many of the same options that are used through the use of special "G" codes or possibly a switch on the control rather than a push button.

Fixed or canned G code cycles (G81–G89) were developed, as stated, to simplify the programming of hole-making operations with repetitive motions at each hole location. These codes are normally modal and must be cancelled with a G80 code (linear rapid transverse mode) before other program movements are commanded. The normal action of the canned cycles mode is to position the programmed nonspindle (tool penetration) axes, including the rotary table axes on milling machines, with the (*Z*) coordinate axis remaining stationary. The tool will then rapid the *Z* axis to the *R* or tool-clearance plane preceding the part surface and feed at the programmed rate to *Z* depth. After feeding the tool to depth, the tool will be retracted from the hole automatically as required by the cycle. Note: different codes will cause various forms of retraction (refer to your programming manual for specific retraction action). Once all of the necessary modal information is programmed (G code, depth, clearance position, spindle speed, feed, and spindle direction), additional holes can be produced by programming new axis or axes positions until the canned cycle is cancelled. Most canned cycles will pick up the last programmed spindle speed/direction and feedrate if they are not initiated in the canned cycle block of information.

Perhaps the most widely used machining sequence is that of drilling a hole, and there are few controls that fail to cater to this requirement by including a canned cycle. Indeed, with word address programming, early attempts were made to standardize a drill cycle. That this was quite successful is evident by the fact that the use of G81 for the purpose is as common as the use of G00 and G01 for linear movement control and G02 and G03 for circular interpolation.

There are a number of machining variations necessary in the production of drilled holes.

One of the most commonly used is the basic drilling movement, catered for by the drilling cycle illustrated in Figure 8.38. This involves a drill movement to the required depth at a controlled feed rate, followed by rapid withdrawal.

Also widely used is the intermittent or "peck" drill cycle for deep holes illustrated in Figure 8.39. This illustration shows a complete withdrawal to the *Z* axis clearance plane after each peck, but variations of this cycle provide for a smaller withdrawal that conveniently breaks the chip but does not give total

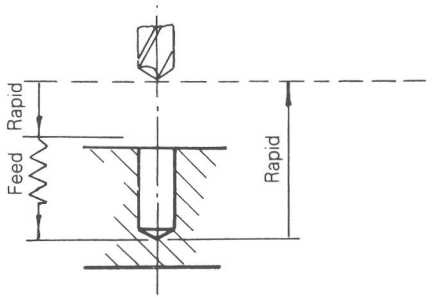


Figure 8.38 *Drill cycle.*

retraction for chip clearance. The peck depth is established with an additional *Z* axis command word, which many times is a "K" value.

A further refinement of this cycle provides for automatic variation of the peck length as the hole deepens. This is achieved by including a "multiplier" in the cycle data. For example, a multiplier of 0.8 will have the effect of reducing each peck length to 0.8 of the previous peck. To avoid the reduction continuing indefinitely a minimum peck length is also programmed.

Closely allied to the drilling cycles are those for boring and tapping. To ensure a clean surface, boring may require special or no drag line retraction, and counterboring may require the inclusion of a time dwell at the extent of cutter travel. Tapping requires that the direction of spindle rotation is reversed to allow withdrawal of the tap.

Many turning operations involve machining chamfer or radius features illustrated in Figures 8.40(a) and 8.40(b). It is possible to program a special cycle on some machines that would normally require two data blocks in just one block. Figure 8.40(a) shows the programming case which would otherwise require a *Z* axis movement followed by simultaneous movement in both the *X* and *Z* axes to produce an angle or chamfer. Figure 8.40(b) shows a similar

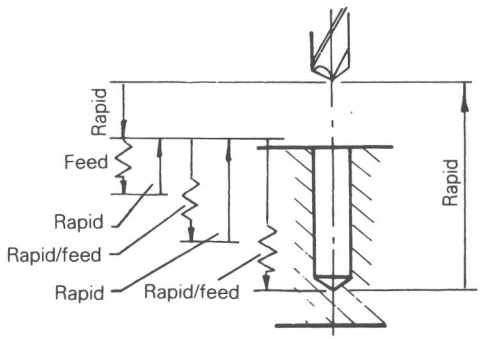


Figure 8.39 *Peck drill cycle.*

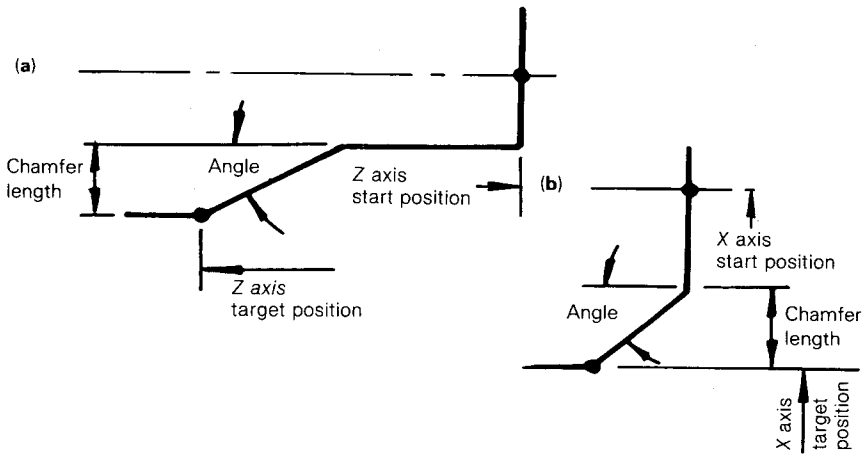


Figure 8.40 *Chamfering cycles.*

situation, but in this case the angular movement is preceded by linear movement in the *X* axis.

Very similar to the cycles described previously are those illustrated in Figures 8.41(a) and 8.41(b). Instead of a linear move being followed by an angular move, the linear moves are followed by radial movement. In both cases the radial movement must be a full 90°.

Both the chamfering and radius cycles may be automatic with little or no programming required for a standard size or there may be special G code and axis information required. Refer to your specific programming manual.

Automatic threading cycles are the counterpart of the G84 tapping canned cycle for single-point threading tools. Single-point thread cutting tools require

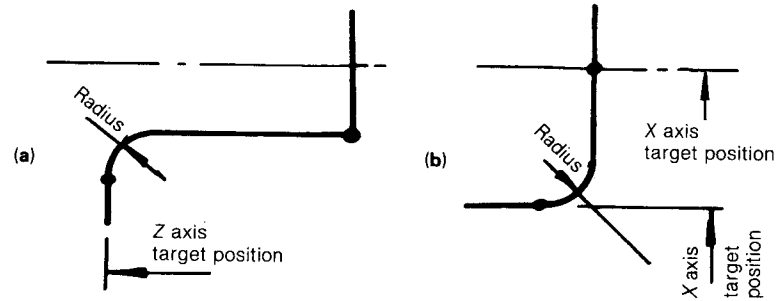


Figure 8.41 *90° arc cycles.*

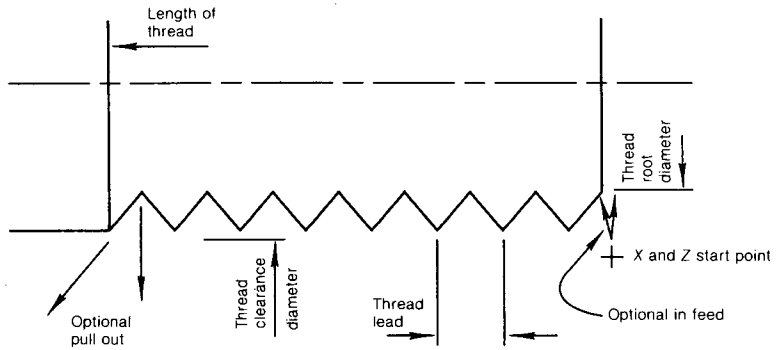


Figure 8.42 *Automatic threading cycle.*

a number of passes to produce a finished thread. Special G codes, like G33, have been developed for lathe work, which allows the programmer to call thread approach and depth, cut distance, retract to clear part, and return to start point all in one block/line of information. Various controls will handle this differently, some requiring a line of instruction for each thread pass and others allowing the entire thread cutting operation to be cut with one statement. G codes like G28 and G29 may be used to define the basic thread pass for later recall. The various codes have been developed to handle both internal and external threads as well as straight or tapered threads. Figure 8.42 shows an example of the types of program information required to make this cycle work.

USER-DEFINED ROUTINES

Canned cycles cater for the easy programming of machined features that are often required on a wide range of components. But the part programmer is often confronted with a feature that is repeated a number of times on a particular component but is found only on that component or a limited range of components. It is in situations such as this that the facility to devise a special routine for use as and when required is very helpful.

Consider the component detail shown in Figure 8.43. Along the length of the shaft is a series of identical recesses. If there is no facility to write a special program, or routine, to machine these recesses the programmer is faced with the rather cumbersome task of detailing each move necessary to machine one recess and then repeating the data for each of the others.

When preparing a routine to accomplish a specific machining task such as this, the programmer can include any of the available canned cycles that might be appropriate. For example, the profile of the shaft recess referred to could be machined using the cycle that permitted one-block programming of a linear

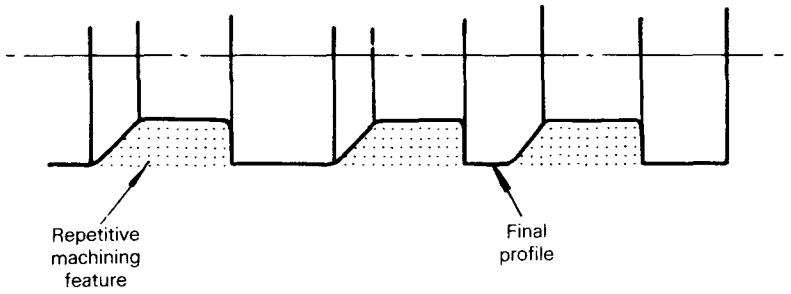


Figure 8.43 *Application of a turning subroutine.*

move in the Z axis folowed by an automatic chamfer or radius. When specific routines are programmed within other routines, they are said to be "nested."

Assume the component shown in Figure 8.44 has a repetitive feature as indicated that justifies using a specially devised routine to clear the recess and mill it to the required profile. Now assume that within each of these recesses there is a series of three smaller recesses as shown in Figure 8.45. Since there will be three times as many smaller recesses as larger recesses, a further specially devised routine to machine them will be justified.

The routine for machining the larger recess will therefore contain the subroutine for machining the smaller recess. The subroutine is nested within the first routine and will be activated three times during the machining of the larger recess.

It would be quite feasible for each of the smaller recesses to include a drilled hole, and this could also be produced using a canned cycle. The subroutine for the smaller recess would now include a nested drilling cycle.

There are usually some limitations regarding nesting. Some controllers permit subroutines within subroutines up to eight deep; others may accept only half this number.

Specially devised routines can also be used to control machine movements and functions not directly associated with metal cutting. For example, a special

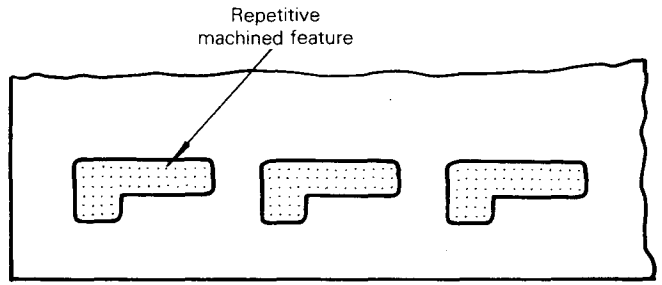


Figure 8.44 *Component detail: application of a milling subroutine.*

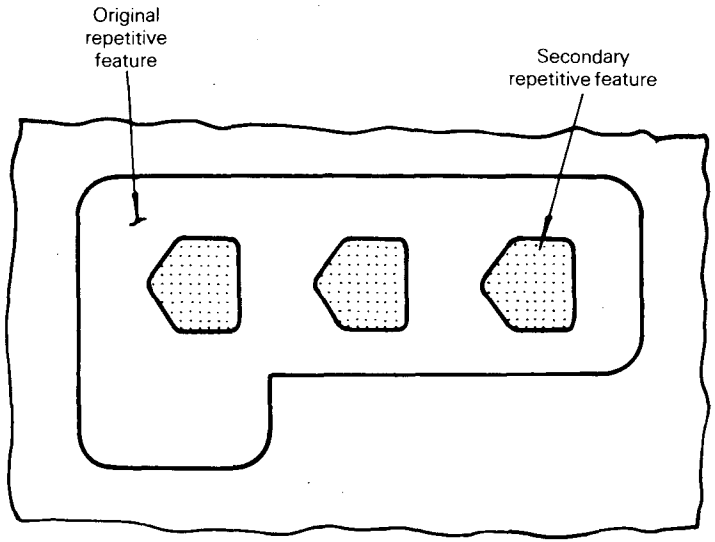


Figure 8.45 *Nesting of subroutines.*

routine can be used to establish and readily recall predetermined parameters relating to machine slide positions and programming modes which may, for safety reasons, need to be established from time to time throughout the program run. The application of a programmer-devised safety routine is included in the sample program listed in Figure 8.46.

The safety routine in this instance is determined by the blocks N10 to N70. It can be seen that, in the program, these data blocks are activated by a G25 program entry. Blocks N100 and N160 are examples where the safety routine is being called. Each time the routine appears in the program the slides return to a safe indexing position and a set of known operating modes is reestablished. This provides a basis from which the programmer can proceed with the programming of further machining operations.

Machine: Hardinge HXL Turning Center Control: GE 1050 Sample Program for Drawing Figure 8.46

N0010	G71	SAFETY ROUTINE	Metric
N0020	G40		Cancels Tool Nose Radius compensation (TNRC)
N0030	G95		Feed mm/rev
N0040	G97 S1000 M03		Spindle rev/min. CW
N0050	G00		Cancels G01, G02, G03, etc.
N0060	G53X177.8Z254 T00		Return to safe indexing position, offsets cancelled
N0070	M01		Optional stop

N0100	G25P$_1$10P$_2$60		Calls safety routine
N0110	T1200		Calls tool
N0120	G54X0Z3T1212		Move in X and Y with zero shift to work face and tool offset active
N0130	S2500F.1	CENTER	Spindle speed and feed
N0140	G01Z-6	DRILL	Center drill
N0150	G00Z2		Drill retract
N0160	G25P$_1$10P$_2$70		Safety routine call with optional stop turret returns to safe indexing position

Note: In the programming areas of special canned cycles and routines, controls vary drastically and your programming manual should be reviewed.

LOOPS

Some control systems provide a "loop" facility. This enables the programmer to devise a routine and to repeat that routine within the part program a specific

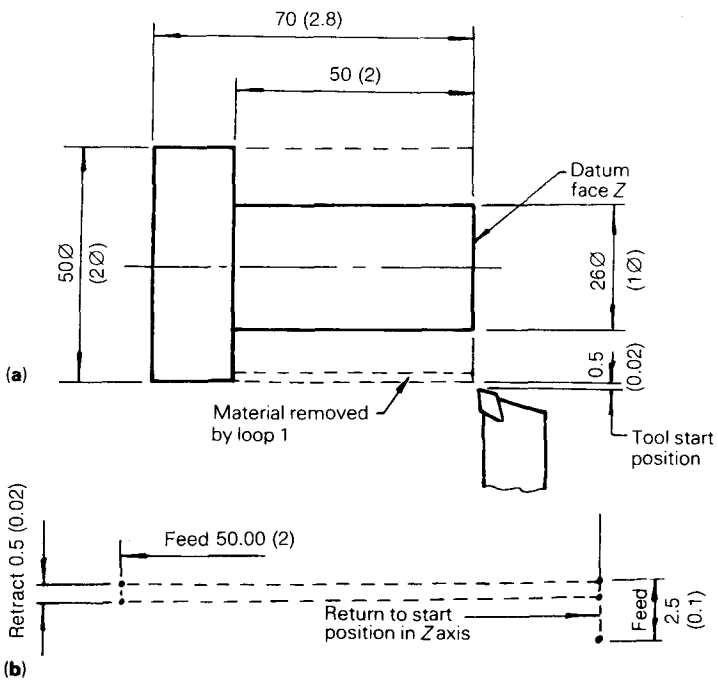


Figure 8.46 *Looping cycle:* (a) *component detail,* (b) *loop detail, repeated six times. (Inch units are given in parentheses.)*

number of times. In other words, when the program reaches the end of the routine the control will return, or loop, back to the beginning of the routine again.

Consider the component shown in Figure 8.46, which is to be reduced from 50 mm (2 in.) diameter to 26 mm (1 in.) diameter by a series of cuts each of 2 mm (0.08 in.) depth. Assuming the starting point for the tool is as shown, the tool will first move in to a depth of 2.5 mm (0.1 in.), thus taking a 2 mm (0.08 in.) depth of cut, travel along a length of 50 mm (2 in.), retract 0.5 mm (0.02 in.) and return to the *Z* datum, so completing a loop. It will then move in a distance of 2.5 mm (0.1 in.), feed along 50 mm (2 in.), retract 0.5 mm (0.02 in.) and return to the *Z* datum, and so on. The loop, including the feed rate, is programmed just once, but is repeated via the loop count data included in the main program as many times as necessary to reduce the work to the required diameter.

MACROS

A somewhat specialized type of programmer-devised routine is referred to as a "macro." This facility can be used for machining a complete component or a feature of a component that, while not standard in the wider sense, is nevertheless frequently required. For instance such a feature may commonly occur within the production schedule of a particular company. The macro is given an identity and stored within a separate macro file, or memory, and is called into use as and when required, possibly as an element within a much larger machining program.

A macro may have fixed dimensions, or it may have parametric variables which enable the dimensions to be varied to produce different versions of a basic component. This technique is referred to as "parametric programming." The parametric or variable version of macro programming is very useful when programming for a family of parts that have the same shape but vary in size.

PARAMETRIC PROGRAMMING

A parameter is a quantity that is constant in one particular case but variable in others. A simple engineering example of a parameter is the length of a bolt. One version of the bolt will have a certain length; all other versions will be identical, that is, they will have the same thread form, diameter, and hexagon head, but they will all vary in length. Thus, the length of the bolt is a parameter, constant in one particular case but variable in others.

Parametric programming involves defining parameters and then using those parameters as the basis for one part program that may be used to machine not only the original component but a number of variations as well.

Figure 8.47(a) shows a component the dimensional features of which have been defined as parameters using the symbol # and a number: #1, #2, #3, and so on.

Figures 8.47(b)–8.47(g) show six variations of the component, the variations being indicated. A range of components such as this is referred to as a family of parts.

The machine movements necessary to machine each of the variations are all included in the original component. Some components require exactly the same movements, but with varying lengths of travel. Other components do not require all of the movements to be made. Using the more usual programming techniques, the production of each component would require a separate part program. Using the parametric part programming technique, instead of defining each dimensional movement individually in the *X* and *Z* axes, the parametric reference is programmed. Thus, to turn along the stepped diameter, the entry in the main program, referred to as the "macro," would read as follows:

```
N07   G01  X#4
N08   Z#2
```

These entries would suffice for all components requiring a stepped diameter. Equally, one entry using parametric identification would suffice for facing all the components to length or drilling the hole.

Having programmed all movements and the sequence in which they are to occur, it remains to define them dimensionally. The dimensional details are entered as a list at the start of the part program. Thus the parameters and their metric dimensional values for the original components would read as follows:

```
#1 = −50.00
#2 = −30.00
#3 = 30.00
#4 = 22.00
#5 = 10.00
```

As each parameter is called in the macro body, the programmed dimensional entry made previously will be invoked.

To machine any of the variations in the family of parts requires a simple amendment of the original parametric values. The parameters to machine the component shown in Figure 8.47(b) would be

```
#1 = −50.00
#2 = −40.00 (amended)
#3 = 30.00
#4 = 22.00
#5 = 10.00
```

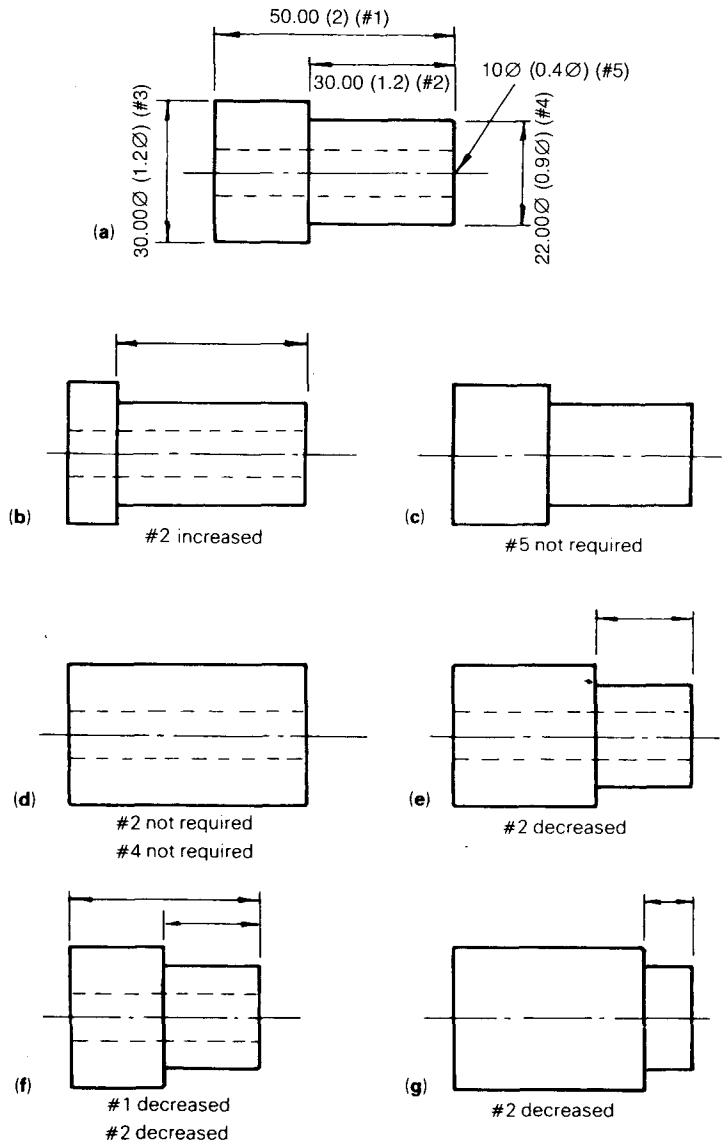



Figure 8.47 *Parametric programming: a family of parts. (Inch units are given in parentheses.)*

and to machine the component in Figure 8.47(f)

#1 = −40.00 (amended)
#2 = −20.00 (amended)
#3 = 30.00
#4 = 22.00
#5 = 10.00

Now consider the components where the programmed movements necessary for machining the basic component are not required. By using a relatively simple programming technique, the control unit can be caused to skip the redundant blocks. The necessary program entry involves the use of certain conditional expressions in which assigned abbreviations are used, such as the following:

EQ = equal to
NE = not equal to
GT = greater than
LT = less than
GE = greater than or equal to
LE = less than or equal to

Consider Figure 8.47(d) and assume the #1 and #3 have been machined. In the macro body the next call will be to machine the stepped diameter. To avoid this, blocks must be skipped and so an entry in the macro body will read as follows:

N15 IF [#4 EQ 0] GO TO N18

This statement says that if #4 is zero, move on to block number 18. Since #4 is nonexistent in the component, the parametric value will be entered as zero and, consequently, the control unit will move ahead.

The preceding description of the use of the parametric programming technique is a very simple one. It is in fact a very powerful concept and its full application is quite complex. For instance, parameters may be mathematically related within the macro body, that is, they may be added together, subtracted from one another, and so on.

In addition, the parametric principle may be extended to include speeds and feeds, when all the likely variations for roughing, finishing, etc., may be given a parametric identity and called into the program as and when required. *Note:* parametric and macro programming, if available on particular machinery, very drastically in programming techniques, so consulting the programming manual is required.

POINT DEFINITION

Point definition is a programming facility, not widely available, which simplifies programming for drilling operations. With this facility it is possible to

define dimensionally as many as 99 points or positions, and enter them into a special file within the control memory. The file can be accessed as required, the points' positions appearing in tabular form.

The points required for inclusion are entered at the start of a part program, and might look as follows:

N1	G78	P1	X15	Y20
N2	G78	P2	X20	Y20
N3	G78	P3	X50	Y30
N4	G78	P4	X65	Y60
N5	G78	P5	X75	X75
N6	G78	P6	X98	Y78

To drill a hole to a specified depth of 20 mm, using a G81 drilling cycle at points 2, 5, and 6, would require a program entry as follows:

N095	G81	Z-20	F150	S1850
N100	G79	P2	P5	P6

The more holes to be drilled, the more advantageous the use of the facility becomes. The dimensional data relating to each point can be modified to suit any particular job.

MIRROR IMAGE

Mirror image is the term that describes a programming facility used to machine components, or features of components, that dimensionally are identical but geometrically opposite either in two axes or one axis. By using the mirror image facility such components can be machined from just one set of data.

In Figure 8.48 an original component feature is indicated in the bottom left-hand corner of the diagram. A complete mirror image of that feature is shown in the top right-hand corner, while mirror images in the *X* axis and the *Y* axis are shown, respectively, in the bottom right- and top left-hand corners of the diagram.

To produce a complete mirror image both the *X* axis and *Y* axis dimensional values will change from negative to positive. For half mirror images the dimensional values will change from negative to positive in one axis only. Mirror imaging is normally achieved by repeating a series of program instructions after activiting a *X* and/or *Y* axis switch or button on the machine controller.

TOOL OFFSETS

Most machining operations involve the use of more than one tool, and usually they vary in length and/or diameter. To accommodate these size variations,

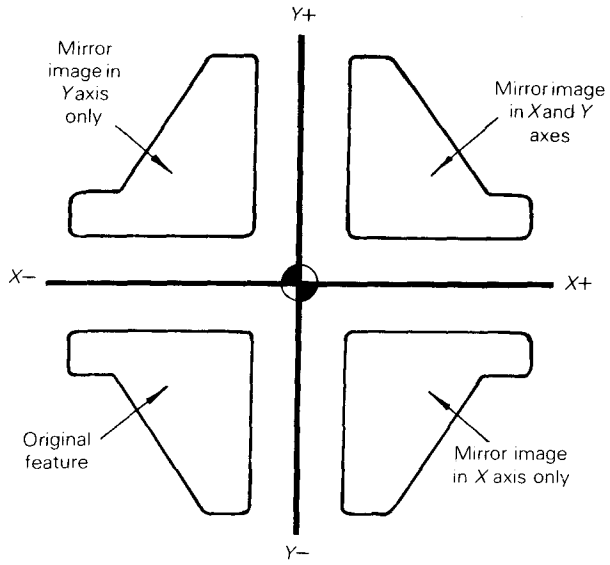


Figure 8.48 *Mirror image.*

and to permit the programmer to assume that all the tools are identical, machine controls are provided with a cutter compensation facility that will, when activated, automatically adjust the programmed slide movements. Thus, it enables the programmer to totally ignore tool size and simply program movements that are exactly the same as the profile detail, a technique referred to as "point" programming or zero-gage-length programming. Some programmers do not agree with this form of programming owing to the inherent dangers if a compensation is missed or gage length incorrectly measured. Therefore, in gage length programming techniques where tool length and radius/diameter are allowed for in program calculations, the tool compensation is used for variations in setup dimensions versus programmed tool lengths.

The task of dealing with variations in tool size is left to the tool setter or operator, since it is essentially a case of ascertaining the tool sizes or size variations and entering these numerical values into the machine control.

The numerical values that are required to be entered relate to tool length and tool radius/diameter.

The manner in which tool length variations are determined and entered varies with machine type. On some controls they are entered simply as offset values, one tool being used as a reference tool and thus having no offset value, and all other tools having data entries corresponding to their dimensional variation from the reference tool. This principle is illustrated in Figure 8.49.

On other machines the size variations of all tools are determined in relation to a fixed point on the machine, such as the corner of a tool post, as illustrated

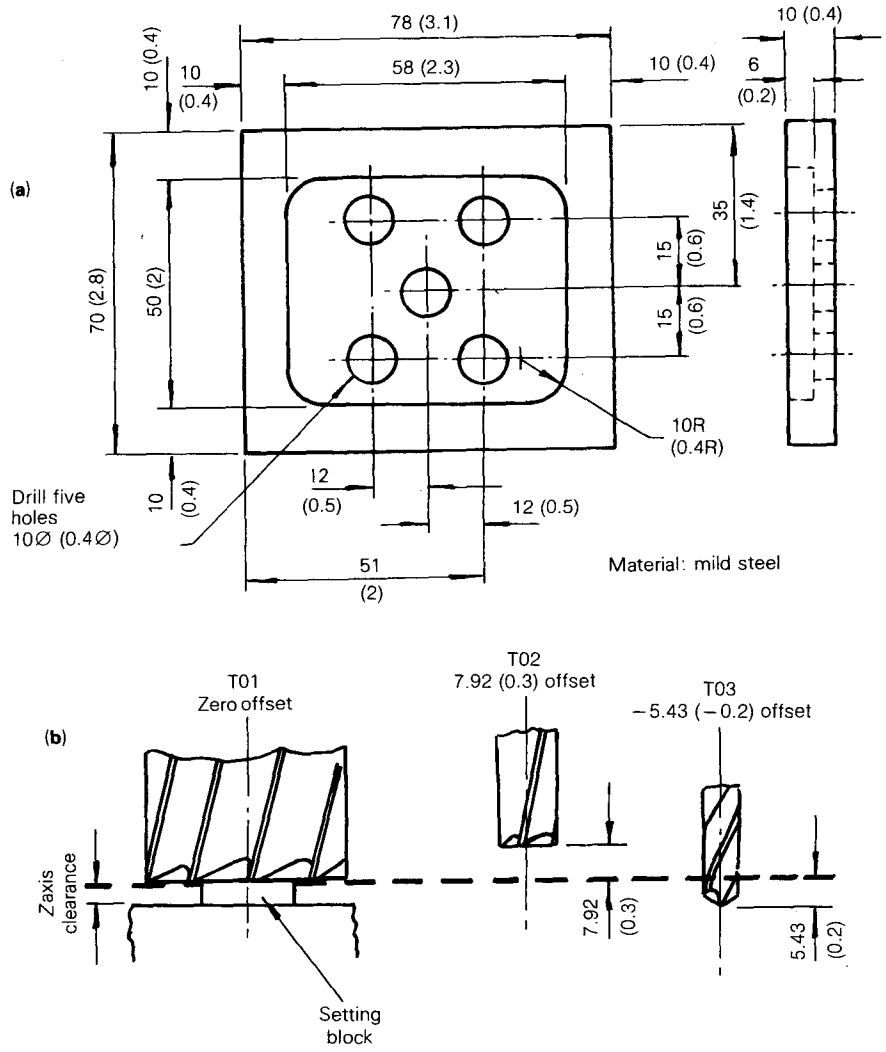


Figure 8.49 *Tool offsets related to a reference tool:* (a) *component detail;* (b) *tool offsets. (Inch units are given in parentheses.)*

in Figure 8.50. In this particular example the variations are entered in a tool data file, there being a second file for programmer-devised offsets that are, through appropriate program data, paired with the tool data file entries.

Tool radius and diameter entries are less complex, being simply a case of ascertaining the correct value, by actually measuring the tool if need be, and

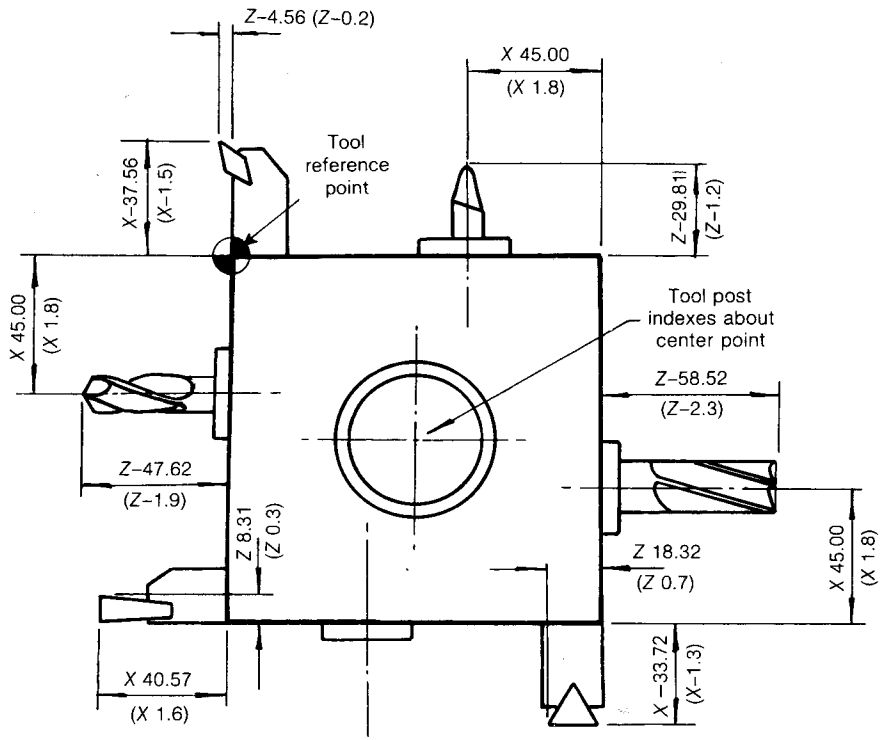


Figure 8.50 *Tool offset data related to a fixed point on the toolpost. (Inch units are given in parentheses.)*

entering the numerical value as required by the control system of the machine being used.

Since tool data can be entered, modified, or erased by the machine operator at will the facility can be used to

(a) accommodate replacement tooling that varies from the original;
(b) make variations to the component size;
(c) initiate a series of cuts, say roughing and finishing, using the same dimensional programmed data.

If the machine operator varies the tool offset values for whatever reason, the effect is temporary. It has no permanent effect on the original program. However, the programmer can utilize the offset facility within the part program, and this creates a situation where accurate communication between the programmer and the shop floor personnel is essential if the programming objectives are to be clearly understood.

Two situations involving tool offsets that are particularly useful from a programming point of view are when variations in the sizes of components are to be made from the same basic program and when a series of cuts are to be initiated along a profile using the same programmed dimensional data.

The ability to use offsets in this manner is based on pairing offset values with specific tools. Just as tools are allocated a numerical identity, so are offsets. Two digits are commonly used, just as in the case of tool identity. The offset digits are paired with the tool digits and are included in the program as part of the tool call. Thus, tool number three with offset number six would be entered in the program as T0306. Older word address programming systems may have special letter codes for calling the tool offset active, such as "H" for length and "D" for diameter.

Control systems usually provide for more offset entry capacity than there will be tools available, so it is possible to call any offset with any tool. The ability of programming any offset with any tool also allows programming more than one offset per tool. An example when to use this feature is when one tool is used to create more than one critical dimension.

The technique of using a number of offsets to make a series of cuts along a profile is illustrated in Figure 8.51. (A similar technique can be used when milling profiles and this is discussed later in the text.)

While the use of offsets as described previously is a very useful programming facility, it should be remembered that the prime objective of an offset facility is to make zero-gage-length programming a possibility and to allow for variation in cutter size during replacement simplifying the programming process. The preceding text has dealt only with tool lengths. It is now necessary to consider the way in which a variety of cutter diameters and tip radii can be accommodated.

To facilitate zero diameter programming with a variety of cutters of varying radii, the control should move the cutter away from the work profile a distance equal to its radius. This facility is referred to as "cutter radius compensation" or "cutter diameter compensation." This compensation also allows for variation in size do to cutter sharpening or deflection.

The distance the cutter will actually move away from or toward the profile—the offset—will be related to the data entry made by the machine setup person or operator. The offset is activated via the appropriate program entry.

The offset can be programmed to occur to the right or left of the required profile, commonly by the use of G41 or G42 when programming in word address mode. To determine which offset code should be programmed, the technique is to imagine being the tool and facing the direction of tool travel. The tool can then be visualized as being either to the right or left of the profile. It is very important to ensure that the correct offset is programmed since a move in the wrong direction may have disastrous results, particularly when large diameter cutters are being used. Tool radius to the right and left of a profile is shown in Figure 8.52.

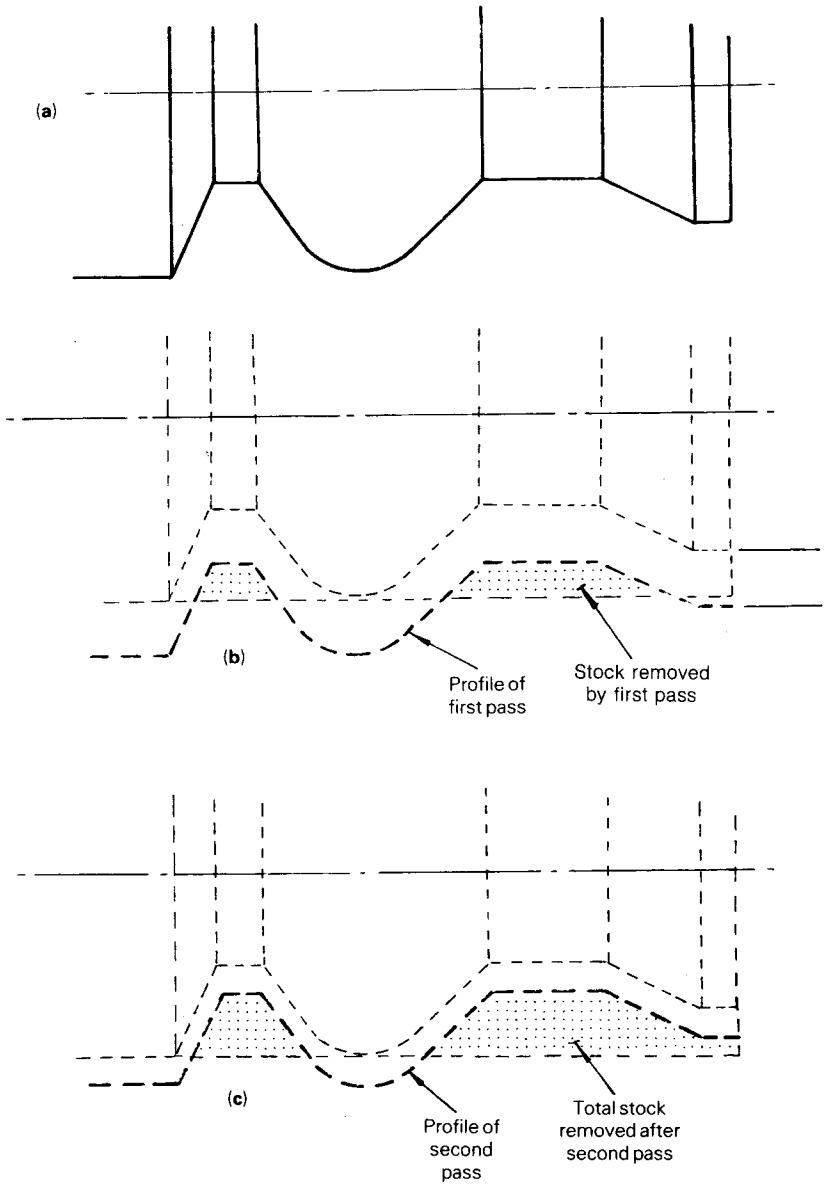


Figure 8.51 *Use of tool offsets for progressive stock removal to a set profile:* (a) *profile detail;* (b) *first cut;* (c) *second cut.*

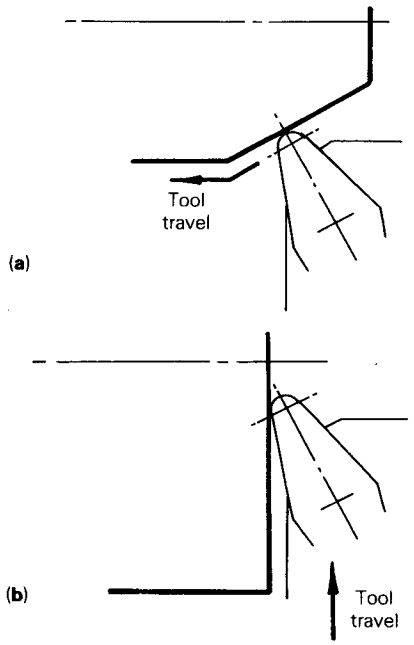


Figure 8.52 *Tool nose radius compensation (TNRC):* (a) *radius compensation left;* (b) *radius compensation right.*

When activating cutter radius compensation, it must be ensured that the slides will first make a noncutting move, to enable the correct tool and workpiece relationship to be established. A similar move is necessary prior to cancellation of the radius compensation. These noncutting moves are referred to as "ramp on" and "ramp off," respectively.

It is now possible to return to the technique referred to earlier, of using offsets to make a series of passes along a milled profile. It is achieved by simply entering a bogus value for the cutter diameter into the control system. By making an entry that is greater than the size of the cutter being used, the actual offset activated via the program will be greater. Thus, the final profile will remain oversize as illustrated in Figure 8.53. The technique can also be used progressively to remove surplus material before making a final cut.

The reverse application of this technique, that is, entering a value smaller than the actual cutter size, will result in a smaller offset and, in the case illustrated, an undersize component profile. Thus, it is possible to produce components of varying dimensions from the same program when milling, just as it is possible to do so when turning.

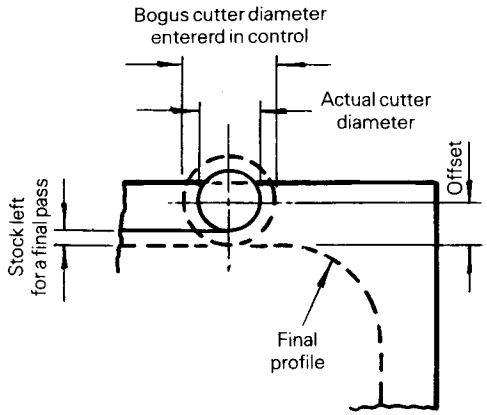


Figure 8.53 *Using an offset to create an oversize milled profile.*

BLOCK DELETE

Production engineering often involves machining a range of components that have slight variations from each other. For example, a hole that has to be drilled in one component is not required in a second component, although all the other details remain the same. Thus, one program would serve for both components providing that some means exists for not drilling the hole when it is not required. The way this is achieved is by use of a "block delete" facility.

Blocks relating to machining features that may not always be required incorporate the symbol /, which is referred to as a slash. The exact position of the slash within the block may vary from one system to another, but it is usually at the start before the sequence number (/N0245).

The machine operator will need to be instructed as to whether the data are to be retained or deleted from the current machining task. If the data are to be retained, the operator takes no action. If the data are to be deleted, the operator has to activate the block delete button on the control panel before running the program. Activation of the button is usually indicated by a light.

If the slash delete button is not activated, the control will respond to all the data contained in the program. If the slash delete button is activated, then all the blocks containing a slash will be ignored.

On some control systems if the slash delete button is not activated, the program will automatically stop when the first slash is reached and the operator then has to make a positive response, either to activate the data contained within the slashes or delete them.

The block delete facility is also useful when machining castings or forgings, where stock removal requirements may vary. The operator is given the option to include an extra cut or not as necessary.

The use of the block delete facility relies on a clear and concise relay of instructions between the part programmer and the machine operator. The machine operator must be left in no doubt as to what is required.

PROGRAM STOPS

Apart from the program stop that is automatically effected when the end of a program is reached, and which arrests all slide and spindle motions, there are two other situations where a halt in proceedings may need to be included in the part program.

The first of these is the point at which the operator is required to carry out some specific task directly associated with the machining program, such as resetting the work or replacing a tool. With word address programming this is normally achieved by programming M00. When such a stop is effected, it is essential that the operator knows exactly what has to be done before the program is reactivated. Some controls will allow for a man readable message to be read from the program and placed on the screen.

The second type of program stop is used when a halt in activity is not quite so critical, and the operator decides whether a stop is actually made. This type of stop is referred to as "optional" and will only take place if the operator has activated the optional stop button on the control console. The programmer may include an optional stop in the program whenever he or she considers it may be of value to the machine operator, such as when a dimensional check or an inspection of the tooling condition is appropriate. But quite often it is the operator who will edit into the program, via the control console, stops that will permit the solving of particular problems that have presented themselves during the machining process. The optional stop in a word address program is normally effected via a programmed M01.

In addition to the stops included in the part program, the operator has, of course, recourse to an emergency stop should the need arise.

CALCULATIONS

It could be argued that, in a well-organized CNC machining environment, the people responsible for the production of the detail drawings of the components to be machined should appreciate the needs of the part programmer and ensure that the drawings are dimensioned accordingly. For example, it is of considerable help when positional slide movements are to be programmed in absolute mode if all dimensions on the drawing are given in relation to a suitable datum. This is especially valuable when the programming technique involves conversational or manual data input, since the last thing a programmer wants to do is to interrupt his thought process in order to calculate unspecified dimensions.

However, whatever the ideal situation may be, it is almost certain that eventually the part programmer will be confronted with a detail drawing that does not cater to his or her requirements. He or she will then find it necessary to make calculations and add dimensions, and perhaps in some cases to completely redimension the drawing.

(The reader should differentiate between poor industrial practice and situations with which he or she may be deliberately confronted in a learning situation, where the objective will be to provide an understanding of the problems likely to be encountered in practice.)

Mention has already been made of the need for dimensions to be given in relation to a set datum when absolute programming is to be used. The opposite situation could also arise, whereby the dimensions are stated in relation to a datum but the programmer needs to program incremental slide moves. In this case the stated dimensions will need to be subdivided. The programmer should exercise caution in this particular situation, and ensure that such an approach is acceptable from a design point of view; minor errors on each of a series of incremental moves could, due to inaccuracies in calculations or round off, accumulate into a larger error that would be unacceptable.

There are other situations that are more complex than simply converting absolute dimensions to incremental and vice versa. Two of these in particular are the need to determine:

(a) profile intersection points;
(b) the location of arc centers.

However complex the profile or shape of a machined surface may appear to be, it can be broken down and defined geometrically as a number of intersecting straight lines or arcs or a combination of the two. To program appropriate machine slide movements, the programmer is required to determine this geometry, and translate production of the profile into a series of linear or circular movements.

Thus, to finish machine the profile shown in Figure 8.54 the following movements will be necessary:

Move 1 Linear
Move 2 Circular, clockwise
Move 3 Linear
Move 4 Circular, counterclockwise
Move 5 Linear

If word address programming is being used, these moves can be described using the appropriate G code: G01, G02, G01, G03, and G01. It may be helpful to mark the drawing accordingly, as in the illustration.

The reader will already appreciate that when programming positional moves, whether they are linear or circular, the target position has to be numerically defined. In this particular example, because the component is dimensioned correctly and the arcs are conveniently 90°, the target position, that is, the intersection points of the geometrical elements of the profile, are readily discerned. No further calculations are necessary.

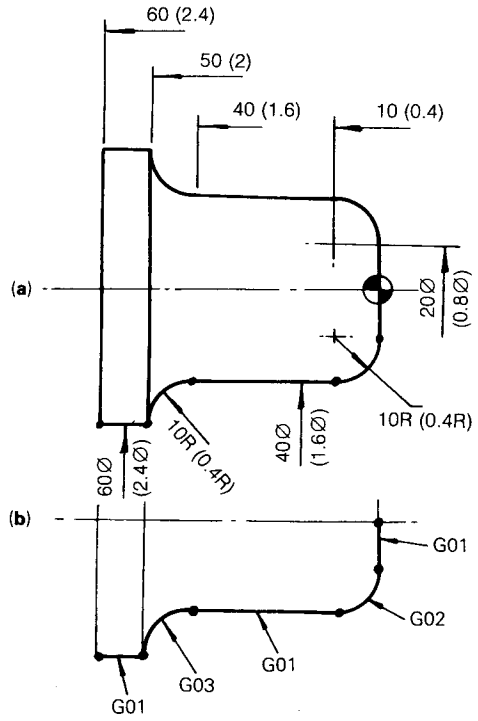


Figure 8.54 *Geometric elements of a profile:* (a) *component detail;* (b) *profile definition. (Inch units are given in parentheses.)*

Consider now the component shown in Figure 8.55(a). Although it is a relatively simple profile, from a programming point of view the drawing is not as helpful as it might be. The target position of the arc, that is, the point at which the circular move ends and the linear move commences, is not defined. Calculations are required as follows in order to determine the target position in the X and Z axes. (Only the metric unit calculation is given.)

In Figure 8.56:

$$\varnothing X_1 = 2(CB + 10) \text{ and } Z_1 = CD$$

In triangle ABC:

$$\angle ABC = 15° \text{ and } AB = 12.5$$

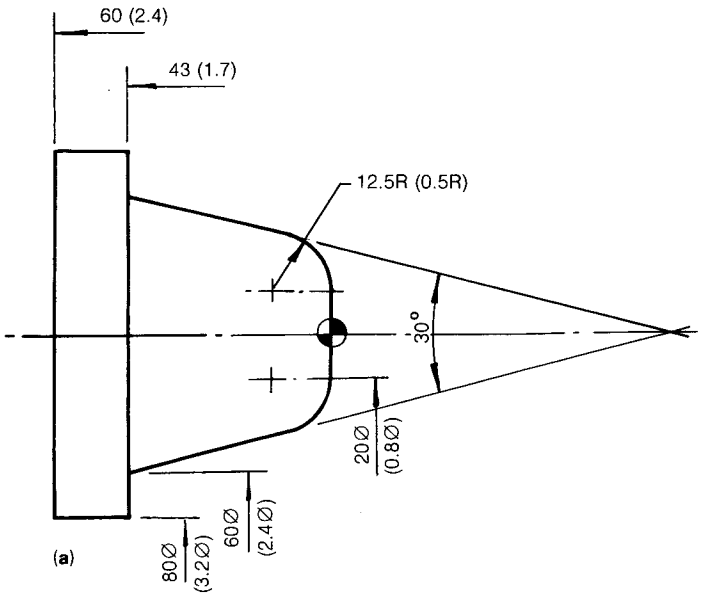


Figure 8.55 *Component detail. (Inch units are given in parentheses.)*

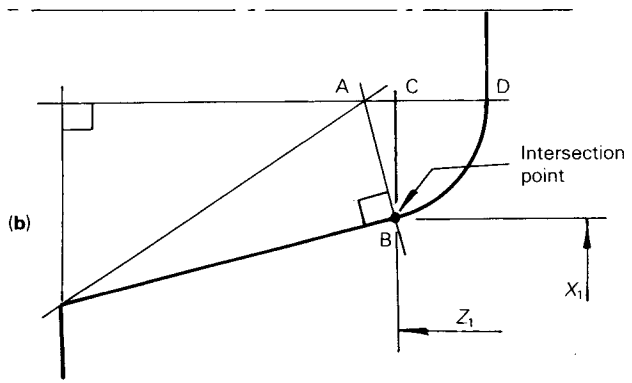


Figure 8.56 *Profile intersection calculation.*

1. To calculate CB:

$$\cos \angle ABC = \frac{CB}{AB}$$

$$\begin{aligned} CB &= \cos \angle ABC \times AB \\ &= \cos 15 \times 12.5 \\ &= 0.966 \times 12.5 \\ &= 12.074 \end{aligned}$$

$$\varnothing X_1 = 2(12.074 + 10) = 44.148$$

2. To calculate AC:

$$\sin \angle ABC = \frac{AC}{AB}$$

$$\begin{aligned} AC &= \sin \angle ABC \times AB \\ &= \sin 15 \times 12.5 \\ &= 0.259 \times 12.5 \\ &= 3.235 \end{aligned}$$

$$\begin{aligned} \text{Since } Z_1 = CD, \text{ then } Z_1 &= AD - AC \\ &= 12.5 - 3.235 \\ &= 9.265 \end{aligned}$$

Thus, the target position, with the *X* value being programmed as a diameter, is X44.148 Z−9.265. These values would need to be included in the part program.

This particular calculation is fairly typical of the situations the part programmer has to deal with. A similar situation presents itself in the profile shown in Figure 8.57(a) where one radius blends with another. The problem is: where does one radius end and the second one start? Calculations are necessary to determine the location of point P in the *X* and *Y* axes as indicated in Figure 8.57(b). The reader may like to consider the solution to the problem. (Answers: 47.81 mm and 71.25 mm, respectively.)

This type of profile also presents the second type of calculation referred to earlier, namely, determining the location of arc centers.

From the previous information the reader will recall that when circular arcs are programmed using word address programming one of three techniques may be involved. All require the target positions to be identified, but the radius definition varies. The first involves defining the arc center in relation to the program datum, and the second requires the arc center to be defined in relation to the arc starting position.

Using the first method, the center of the 30 mm radius arc is easily determined from the dimensions already on the drawing. But the location of the center of the 50 mm radius is not so straightforward and a calculation is required.

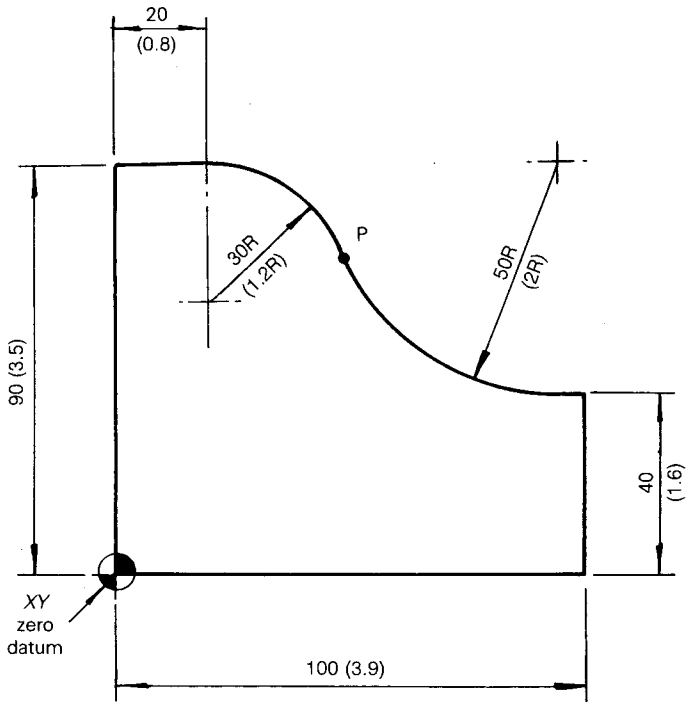


Figure 8.57 (a) *Profile detail. (Inch units are given in parentheses.)*

Using the second method of circular interpolation the definition of the arc centers in relation to the start points again presents a problem as far as the 50 mm radius is concerned, and a calculation will be necessary before the program can be written.

Exercises involving the calculation of profile intersection points and arc centers are included in this chapter's section titled "Part Programming Calculations."

TOOL PATHS

A prime objective of the part programmer should be to ensure that a component is machined in the shortest possible time. Earlier in the text reference was made to the way a well-planned sequence of machining operations can contribute to this objective. But often within each individual machining sequence there is room for further efficiency, which results in time saving.

Consider the drilling of the series of five holes in the component shown in Figure 8.58. Two sequences in which the holes might be drilled are indicated. Which sequence would be the quicker?

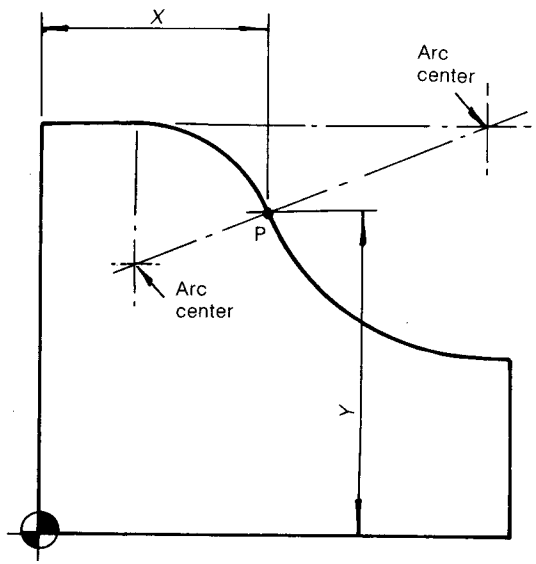


Figure 8.57 (b) *Required profile intersection dimensions.*

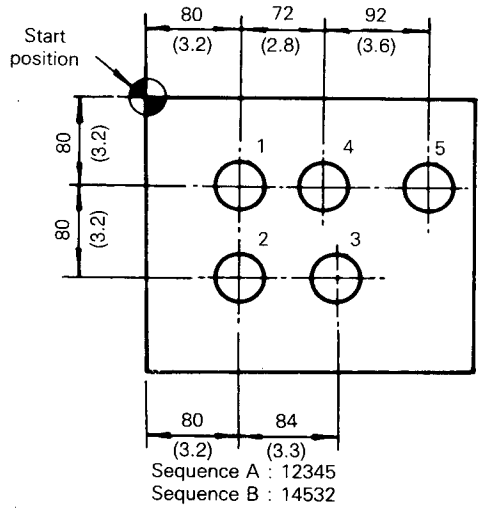


Figure 8.58 *Alternative drilling sequences. (Inch units are given in parentheses.)*

The actual operation of drilling the holes, that is, movement in the Z axis, would be identical in both cases. Therefore, any saving that can be achieved must be by reducing the total length of the positioning moves, and therefore the time taken.

Providing the detail drawing is reasonably accurately drawn a simple rule measurement check may suffice to determine the shorter route. Applying this technique to this particular example will reveal that the second sequence is quicker than the first.

The need to give careful consideration to tool paths is also important during stock removel operations. This is particularly so when there are no stock removal canned cycles available within the control system, or if they cannot be utilized in a particular situation.

Consider the removal of stock, or area clearance as it is also known, in order to machine the step shown in Figure 8.59.

If a pocket milling cycle is available on the control system of the machine, this could be used, the missing sides of the "pocket" being indicated by the dotted line. Use of the cycle would ensure that efficient tool paths are employed.

If such a cycle is not available, then the matter becomes a little more complex. The process of producing the step will involve programming a series of linear moves, with careful attention being given to providing an appropriate cutter overlap to ensure a clean face. The lengths of relative cutter travel will also have to ensure a uniform amount of metal is left for a finish pass along

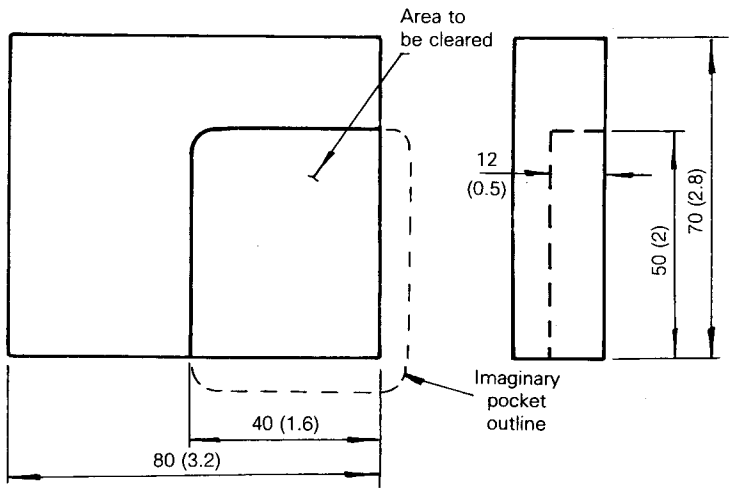


Figure 8.59 *"Pocket" detail. (Inch units are given in parentheses.)*

the profile. The programmer should also ensure that the cutter paths used are the shortest and therefore the quickest.

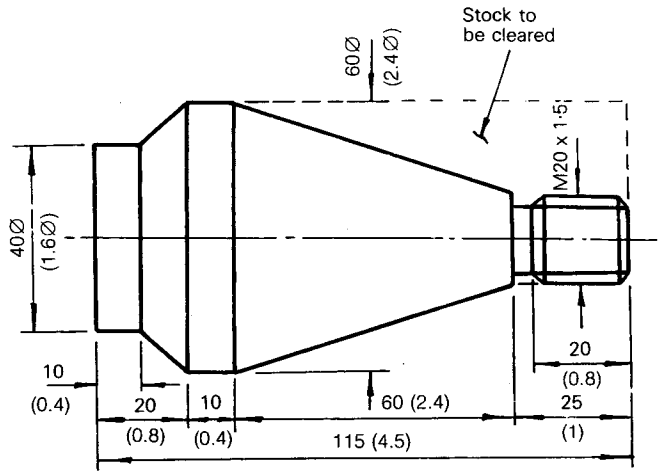
Similar problems often present themselves during turning operations. Figure 8.60 shows a typical example.

There is no short cut when solving this type of problem. After a little experience of dealing with situations of this nature, the trainee programmer soon comes to appreciate the value of canned cycles, which reduce the amount of machining dealt with in this manner to a minimum.

If the part programmer is confronted with machining situations such as these, he or she will have to resort to drawing the profile, preferably to an enlarged scale, and then imposing appropriate tool paths on the drawing. In the case of milling examples it may be necessary to draw circles indicating the cutter diameter. The milled step referred to above, when dealt with in this way, is shown in Figure 8.61. Having decided on the most suitable tool paths (which may take a number of attempts), the slide movements may be dimensionally determined by carefully scaling the drawing or through mathematical calculation.

An alternative approach is to reproduce the profile on graph paper, as shown in Figure 8.62, in which case the graduated lines on the graph paper can be used to determine the dimensional value of the necessary moves.

Exercises involving the determination of cutter paths are included in subsequent examples.



Material: medium carbon steel 60Ø (2.4Ø)

Figure 8.60 *Component requiring excess stock removal. (Inch units are given in parentheses.)*

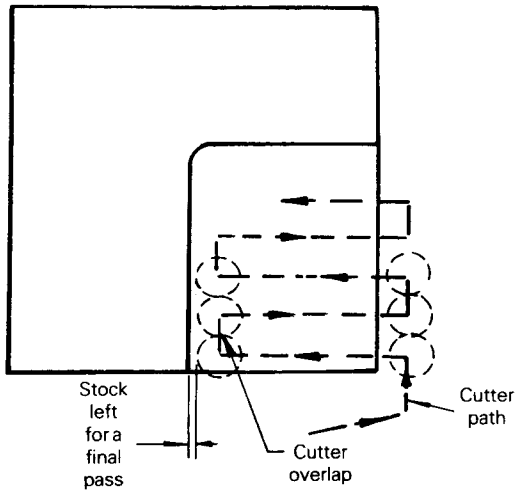


Figure 8.61 *Determination of cutter path to mill a step.*

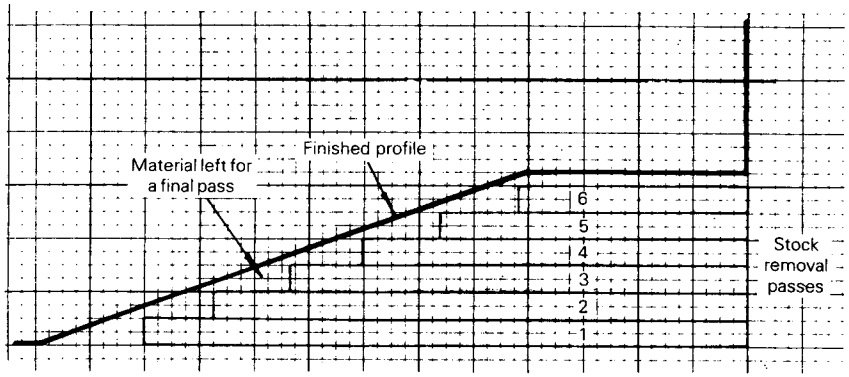


Figure 8.62 *Determination of tool paths when turning.*

PROGRAM LISTING AND PROVING

Before starting to create a part program listing, all the various facets of competent part programming discussed so far should have received due attention.

The sequence of operations, together with the tooling and work-holding techniques to be employed, should be documented. Appropriate speeds and feed rates should have been determined and all the necessary calculations affecting slide movements must be complete.

Having reached this stage the way is clear to list the program, and this requires the programmer to be conversant with the machine programming language.

To become fully proficient with a particular programming system takes time and practice. As with most things it is a case of starting with relatively simple tasks and gradually progressing to more complex examples. If you are a student, or perhaps undergoing training in an industrial establishment, it is almost certain that your course work will be structured in this way.

Competent part programming demands a logical approach and a high degree of concentration and care when actually listing the program. Mistakes are easily made and can have disastrous results, although fortunately most mistakes can be discovered and rectified before machining takes place.

Programs may be listed on appropriate forms, or on plain paper or they can be entered into a computer and listed on the display screen. Programs initially handwritten can, of course, also be entered into a computer and visually displayed.

The use of a computer for program listing is often coupled with the facility to prove the program using animated computer graphics. This involves, in effect, "machining" the component on the screen.

The effectiveness of proving programs in this way will depend on the sophistication of the software available. The simplest software will usually highlight major errors such as movement occurring in the wrong direction or a lathe tool crashing into a chuck, while the more complex will also indicate errors relating to speeds and feeds and even the absence of a coolant supply.

Ultimately, the part program will be entered into the machine control unit, but this may also involve computer graphics. Figure 8.63 shows a controller that includes a built-in visual display. As the program is entered the CRT screen will display the geometric profile of the part and the programmed cutter paths and thus confirm, or otherwise, the validity of the data input. The illustration relates to a program written for the part detailed in Figure 8.64. An enlargement of the CRT display is shown in Figure 8.65 where the component profile can be more readily defined.

A large number of machines currently in use do not have the benefit of built-in computer graphics, and if off-line computer graphics proving facilities are not available, then the proving of the part program must take the form of a test run or a dry run or both.

The test run is basically a check that the data input is valid, that is, that the machine is capable of responding to the data entries included in the program. Data errors are usually indicated by a displayed message. No slide movement takes place during the test run.

The dry run procedure also excludes metal cutting, but, with this checking procedure, slide movements occur at a rapid rate of traverse. This test ensures that the intended machine movements are occurring, and that they will result in the machined features required.

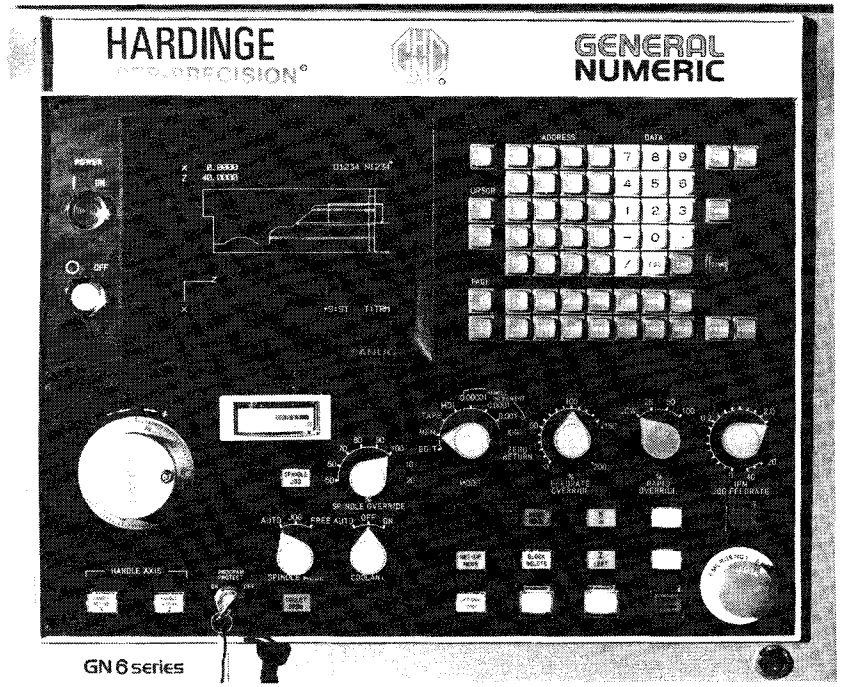
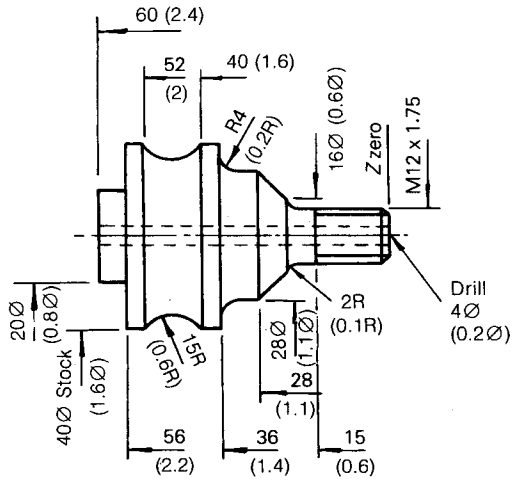


Figure 8.63 *A machine controller with built-in CRT to facilitate program proving.*



Material: aluminum allow 40Ø (1.6Ø)

Figure 8.64 *Component detail. (Inch units are given in parentheses.)*

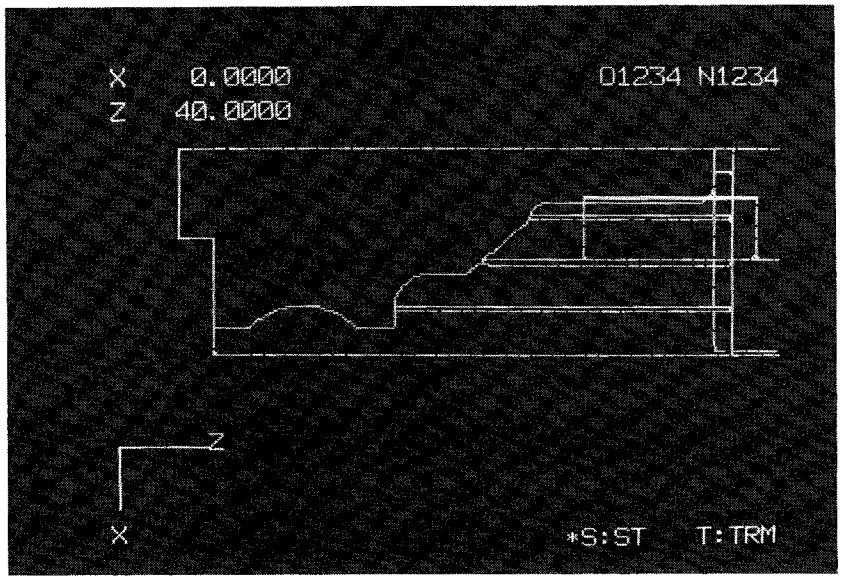


Figure 8.65 *An enlargement of the CRT display shown in Figure 8.63.*

The most common test available, even on the simplest of machines, is to run the program through one block at a time with reduced programmed feed rates, but without the job material being in position so that no machining actually takes place—block by block prove out. This prove out method is usually followed by a block by block run with part material and reduced feed rates before an automatic run is attempted.

Tests of this nature carried out on the machine may or may not be the responsibility of the programmer, although he or she will soon be involved if any errors are indicated.

Finally, there is the need, particularly in industrial situations, to record the program for future use. In its simplest form, storage can be a handwritten version of the proved program. Alternatively, it may be in the form of a perforated tape or be recorded on a magnetic tape or disk.

Whatever the storage medium, it must be remembered that people looking to reuse the program at a later date will also need information relating to tooling and workholding. This information is as critical as the part program and must also be carefully filed for future reference.

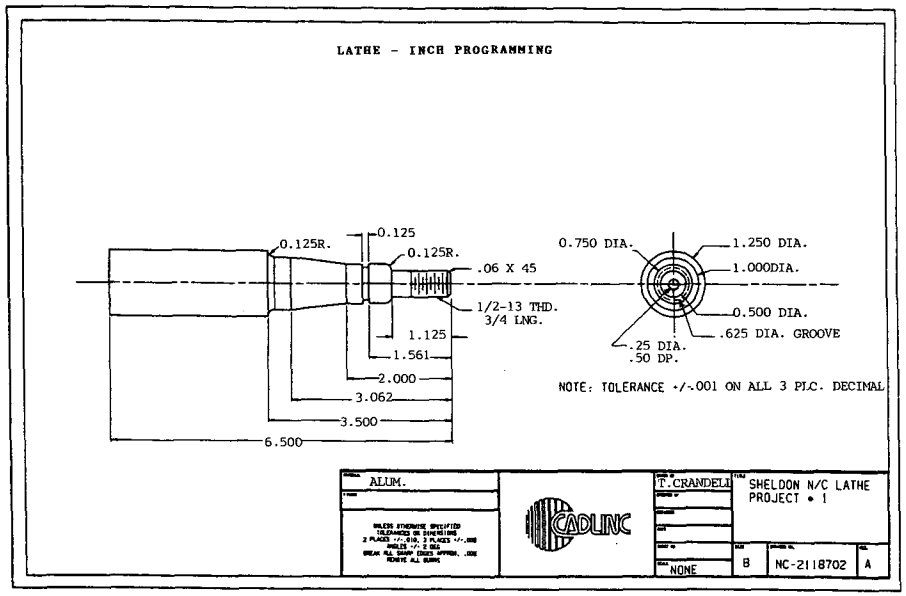
MANUAL PART PROGRAMMING EXAMPLES

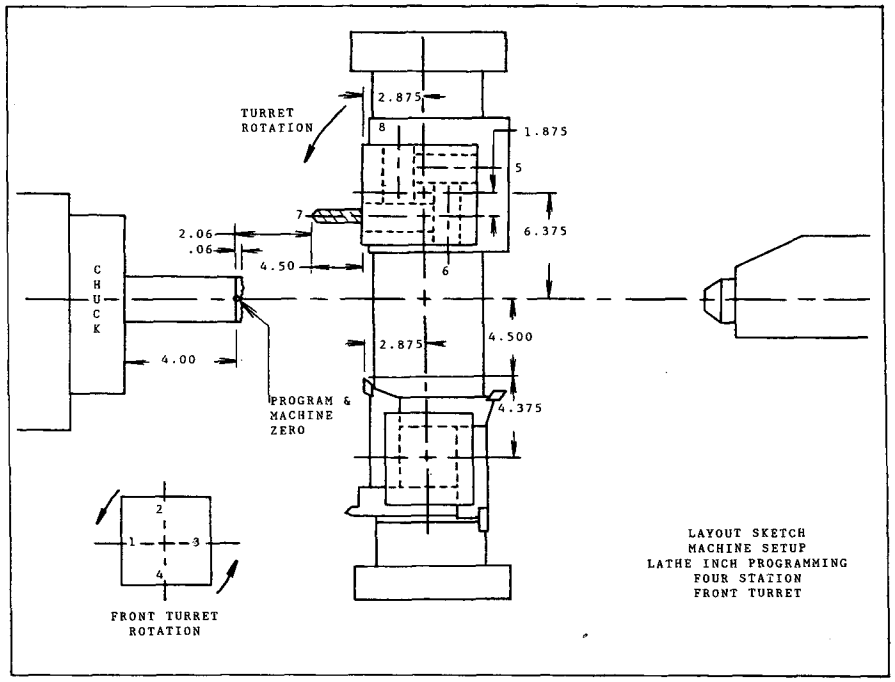
The programming examples that follow were prepared to show the calculation and writing of generic manual part programs. Examples follow for both a lathe

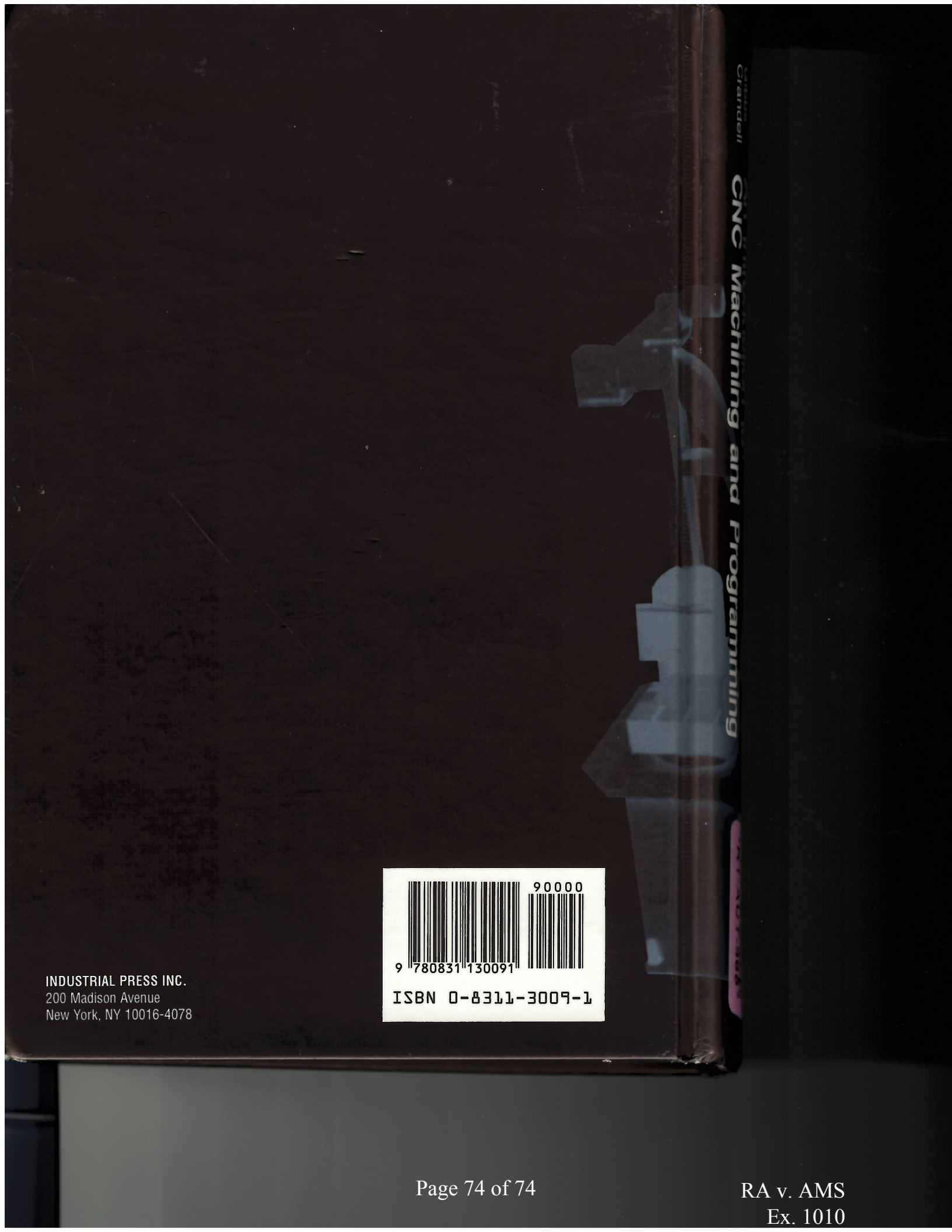
and mill in inch and metric calculations. The intent of the author is to give you the realization of putting an entire program document together. We must realize though that most machines program slightly different and programmers take various approaches to how they process, tool, and program.

Finally, note that it is common practice to number blocks of information by increments of five or ten: N0010, N0020, N0030, and so on. The reason for adopting this approach is that if, on completion of the program, it is found that something has been omitted, it will be possible to insert additional blocks. It also provides space that will facilitate general editing of the program at the machine control should this be found to be necessary.

Example 1: Lathe Inch Programming







INDUSTRIAL PRESS INC.
200 Madison Avenue
New York, NY 10016-4078

90000
9 780831 130091
ISBN 0-8311-3009-1