

Module III

NC PART PROGRAMMING

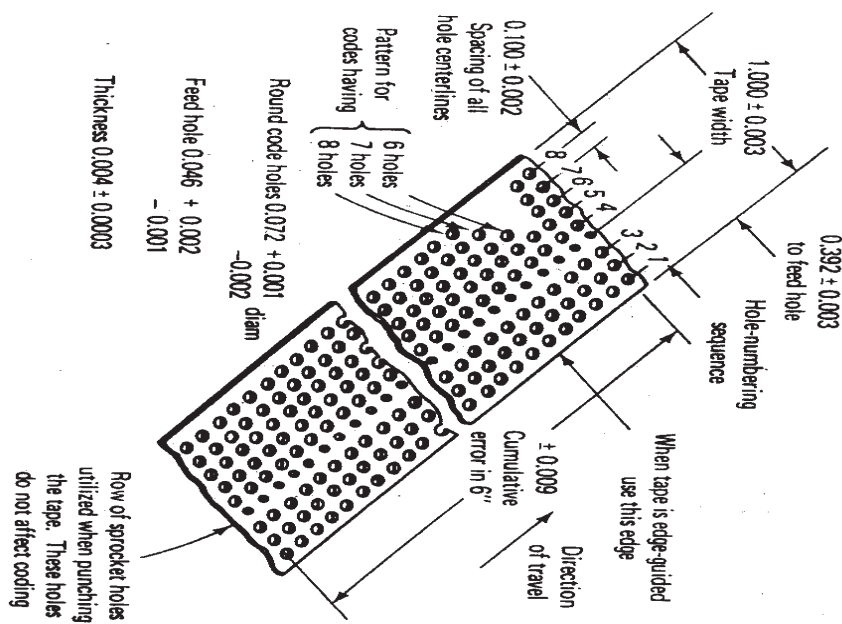
INTRODUCTION

Numerical control part programming is the procedure by which the sequence of processing steps to be performed on the NC machine is planned and documented. It involves the preparation of a punched tape (or other input medium) used to transmit the processing instructions to the machine tool. There are two methods of part programming: manual part programming and computer-assisted part programming. In this chapter we describe both of these methods, with emphasis on the latter.

It is appropriate to begin the discussion of NC part programming by examining the way in which the punched tape is coded. Coding of the punched tape is concerned with the basic symbols used to communicate a complex set of instructions to the NC machine tool. In numerical control, the punched tape must be generated whether the part programming is done manually or with the assistance of some computer package. With either method of part programming, the tape is the net result of the programming effort. In coming sections our attention will be focused on the punched tape and the structure of the basic language used by the NC system.

PUNCHED TAPE IN NC

The part program is converted into a sequence of machine tool actions by means of the input medium, which contains the program, and the controller unit, which interprets the input medium. The controller unit and the input medium must be compatible. That is, the input medium uses coded symbols which represent the part program, and the controller unit must be capable of reading those symbols. [The most common input medium is punched tape. The tape has been standardized, so that type punchers are manufactured to prepare the NC tapes, and tape readers part of the controller unit) can be manufactured to read the tapes. The punched tape used for NC is 1 in. wide. It is standardized as shown in Figure by the electronics Industries Association (EIA), which has been responsible for many of the important standards in the NC industry.



There are two basic methods of preparing the punched tape. The first method associated with manual part programming and involves the use of a typewriter like device. Figure illustrates a modern version of this kind of equipment. The operator types directly from the part programmer's handwritten list of coded instructions. This produces a typed copy of the program as well as the punched type. The second method is used with computer-assisted part programming. By this approach, the tape is prepared directly by the computer using a device called a tape punch.

By either method of preparation, the punched tape is ready for use. During production on a conventional NC machine, the tape is fed through the tape reader once for each workpiece. It is advanced through the tape reader one instruction at a time. While the machine tool is performing one instruction, the next instruction is being read into the controller unit's data buffer. This makes the operation of the NC system more efficient. After the last instruction has been read into the controller, the tape is rewound back to the start of the program to be ready for the next workpart.

TAPE CODING AND FORMAT

NC tape coding

As shown in Figure, there are eight regular columns of holes running in the lengthwise direction of the tape. There is also a ninth column of holes between the third and fourth regular columns. However, these are smaller and are used as sprocket holes for feeding the tape.

Figure shows a hole present in nearly every position of the tape. However, the coding of the tape is provided by either the presence or absence of a hole in the various positions. Because there are two possible conditions for each position—either the presence or absence of a hole—this coding system is called the binary code. It uses the base 2 number system, which can represent any number in the more familiar base 10 or decimal system. The NC tape coding system is used to code not only numbers, but also alphabetical letters and other symbols. Eight columns provide more than enough binary digits to define any of the required symbols.

How instructions are formed

A binary digit is called a bit. It has a value of 0 or 1 depending on the absence or presence of a hole in a certain row and column position on the tape. (Columns of hole positions run lengthwise along the tape. Row positions run across the tape.) Out of a row of bits, a character is made. A character is a combination of bits, which represents a letter, number, or other symbol. A word is a collection of characters used to form part of an instruction. Typical NC words are x position, y position, cutting speed, and so on. Out of a collection of words, a block is formed. A block of words is a complete NC instruction. Using an NC drilling operation as an example, a block might contain information on the x and y coordinates of the hole location, the speed and feed at which the cut should be run, and perhaps even a specification of the cutting tool.

To separate blocks, an end-of-block (EOB) symbol is used (in the EIA standard, this is a hole in column 8). The tape reader feeds the data from the tape into the buffer in blocks. That is, it reads in a complete instruction at a time.

NC words

Following is a list of the different types of words in the formation of a block. Not every NC machine uses all the words. Also, the manner in which the words are expressed will differ between machines. By convention, the words in a block are given in the following order:

SEQUENCE NUMBER (n-words): This is used to identify the block.

PREPARATORY WORD (g-words): This word is used to prepare the controller for instructions that are to follow. For example, the word g02 is used to prepare the C controller unit for circular interpolation along an arc in the clockwise direction. The preparatory word is needed so that the controller can correctly interpret the data that follow it in the block.

COORDINATES (x-, y-, and z-words): These give the coordinate positions of the tool. In a two-axis system, only two of the words would be used. In a four- or five-axis machine, additional a-words and V or b-words would specify the angular positions.

Although different NC systems use different formats for expressing a coordinate, we will adopt the convention of expressing it in the familiar decimal form: For example, x + 7.235 or y - 0.540. Some formats do not use the decimal point in writing the coordinate. The + sign to define a positive coordinate location is optional. The negative sign is, of course, mandatory.

FEED RATE (f-word): This specifies the feed in a machining operation. Units are inches per minute (ipm) by convention.

CUTTING SPEED (s-word): This specifies the cutting speed of the process, the rate at which the spindle rotates.

TOOL SELECTION (t-word): This word would be needed only for machines with a tool turret or automatic tool changer. The t-word specifies which tool is to be used in the operation. For example, t05 might be the designation of a 1/2-in. drill bit in turret position 5 on an NC turret drill.

MISCELLANEOUS FUNCTION (m-word): The m-word is used to specify certain miscellaneous or auxiliary functions which may be available on the machine tool. Of

course, the machine must possess the function that is being called. An example would be m03 to start the spindle rotation. The miscellaneous function is the last word in the block. To identify the end of the instruction, an end-of-block (EOB) symbol is punched on the tape.

MANUAL PART PROGRAMMING

To prepare a part program using the manual method, the programmer writes the machining instructions on a special form called a part programming manuscript. The instructions must be prepared in a very precise manner because the typist prepares the NC tape directly from the manuscript. Manuscripts come in various forms, depending on the machine tool and tape format to be used. For example, the manuscript form for a two-axis point-to-point drilling machine would be different than one for a three-axis contouring machine. The manuscript is a listing of the relative tool and workpiece locations. It also includes other data, such as preparatory commands, miscellaneous instructions, and speed/ feed specifications, all of which are needed to operate the machine under tape control.

Manual programming jobs can be divided into two categories: point-to-point jobs and contouring jobs. Except for complex work parts with many holes to be drilled, manual programming is ideally suited for point-to-point applications. On the other hand, except for the simplest milling and turning jobs, manual programming can become quite time consuming for applications requiring continuous-path control of the tool. Accordingly, we shall be concerned only with manual part programming for point-to-point operations. Contouring is much more appropriate for computer-assisted part programming.

The basic method of manual part programming for a point-to-point application is best demonstrated by means of an example.

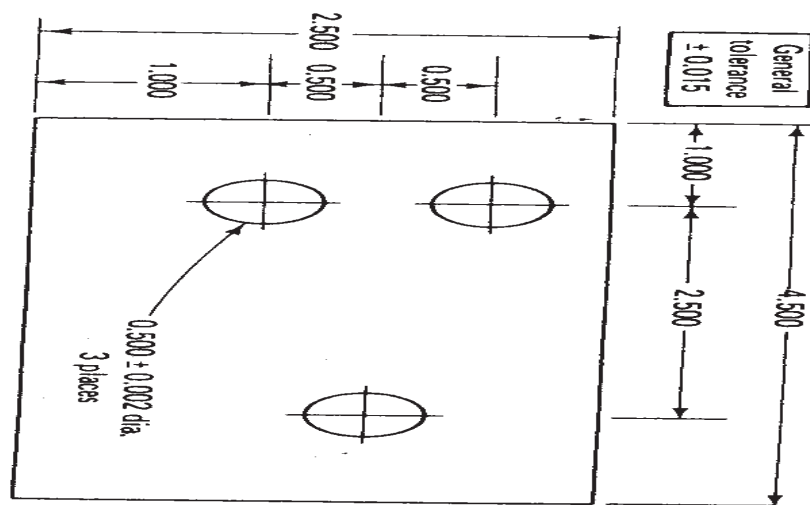
EXAMPLE

Suppose that the part to be programmed is a drilling job. The engineering drawing for the part is presented in Figure. Three holes are to be drilled at a diameter of 31 in. The close hole size tolerance requires reaming to 0.500 in. diameter.

Recommended speeds and feeds are as follows:

	Speed (rpm)	Speed (in./min)
0.484-in.-diameter drill	592	3.55
0.500-in.-diameter drill	382	3.82

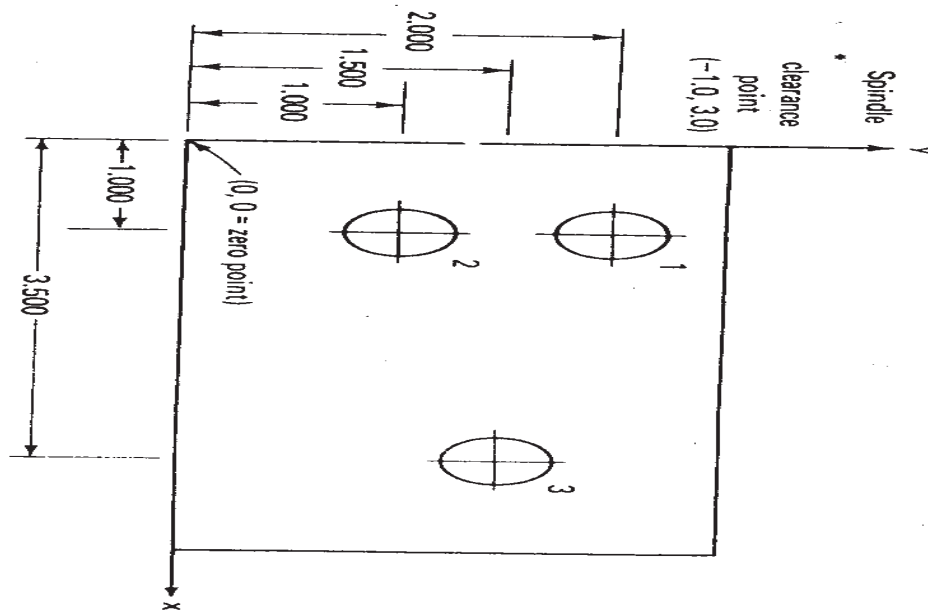
The NC drill press operates as follows. Drill bits are manually changed by the machine operator, but speeds and feeds must be programmed on the tape. The machine has the floating-zero feature and absolute positioning.



Part drawing for Example.

The first step in preparing the part program is to define the axis coordinates in relation to the work part. We assume that the outline of the part has already been machined before the drilling operation. Therefore, the operator can use one of the comers of the part as the target point. Let us define the lower left-hand comer as the target point and the origin of our axis system. The coordinates are shown in Figure for the example part. The x and y locations of each hole can be seen in the figure. The completed manuscript would appear as in Figure. The first line shows the x and y coordinates at the zero point. The machine operator would insert the tape and read this first block into the system. (A block of instruction corresponds generally to one line on the manuscript form.) The tool would then be positioned over the target point

on the machine table. The operator would then press the zero buttons to set the machine.



Coordinate system defined for PART IN example

The next line on the manuscript is RWS, which stands for rewind-stop. This signal is coded into the tape as holes in columns 1, 2, and 4. The symbol stops the tape after it has been rewound. The last line on the tape contains the m30 word, causing the tape to be rewound at the end of the machining cycle. Other m-words used in the program are m06, which stops the machine for an operator tool change, and m13, which turns on the spindle and coolant. Note in the last line that the tool has been repositioned away from the work area to allow for changing the workpiece.

NC part programming manuscript
Two axis point-to-point drill press
tab sequential format

Part No. EXAMPLE 8.2 Date 4/4/79
Part Name MOULD PLATE Prepared by MPC

Seq. No.	Tab EOB	X COORD	Tab EOB	Y COORD	Tab EOB	Feed	Tab EOB	Speed	Tab EOB	m WORD	Tab EOB	Comments
00	TAB	0.0	TAB	0.0	EOB							ZERO
KMS												
01	TAB	1.0	TAB	2.0	TAB	3.55	TAB	592	TAB	13	EOB	DRILL 1
02	TAB		TAB	1.0	EOB							DRILL 2
03	TAB	3.5	TAB	1.5	EOB							DRILL 3
04	TAB	-1.0	TAB	3.0	TAB		TAB		TAB	06	EOB	TOOL CHG
05	TAB	3.5	TAB	1.5	TAB	3.82	TAB	392	TAB	13	EOB	REAM 3
06	TAB	1.0	TAB	1.0	EOB							REAM 2
07	TAB		TAB	2.0	EOB							REAM 1
08	TAB	-1.0	TAB	3.0	TAB		TAB		TAB	06	EOB	TOOL CHG
09	TAB		TAB		TAB		TAB		TAB	30	EOB	RELOAD
												CHG PART

12

Part program manuscript for Example.

COMPUTER-ASSISTED PART PROGRAMMING

The workpart of Example was relatively simple. It was a suitable application for manual programming. Most parts machined on NC systems are considerably more complex. In the more complicated point-to-point jobs and in contouring applications, manual part programming becomes an extremely tedious task and subject to errors. In these instances it is much more appropriate to employ the high-speed digital computer to assist in the part programming process. Many part programming language systems have been developed to perform automatically cost of the calculations which the programmer would otherwise be forced to do. This saves time and results in a more accurate and more efficient part program.

The part programmer's job

Computer-assisted part programming, the NC procedure for preparing the tape from the engineering drawing is followed as. The machining instructions are

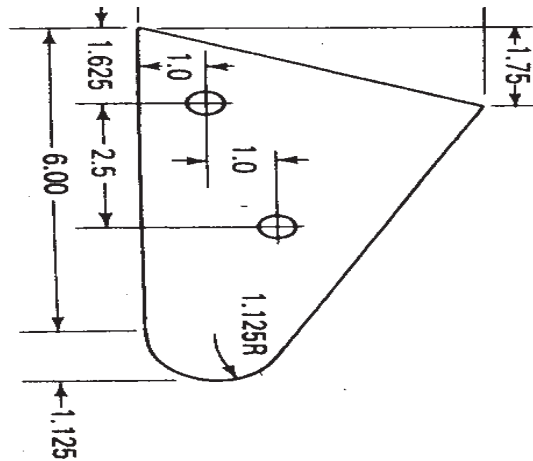
written in English-like statements of the NC programming language, which are then processed by the computer to prepare the tape. The computer automatically punches the tape in the proper tape format for the particular C machine.

The part programmer's responsibility in computer-assisted part programming consists of two basic steps:

1. Defining the workpart geometry
2. Specifying the operation sequence and tool path

No matter how complicated the workpart may appear, it is composed of six geometric elements. Using a relatively simple workpart to illustrate, consider the component shown in Figure. Although somewhat irregular in overall appearance, the outline of the part consists of intersecting straight lines and a partial circle. The holes in the part can be expressed in terms of the center location and radius of the hole. Nearly any component that can be conceived by a designer can be described by points, straight lines, planes, circles, cylinders, and other mathematically defined surfaces. It is the part programmer's task to enumerate the elements out of which the part is composed. Each geometric element must be identified and the dimensions and location of the element explicitly defined.

After defining the workpart geometry, the programmer must next construct the path that the cutter will follow to machine the part. This tool path specification involves a detailed step-by-step sequence of cutter moves. The moves are made among the geometry elements, which have previously been defined. The part programmer can use the various motion commands to direct the tool to machine along the workpart surfaces, to go to point locations, to drill holes at these locations, and so on. In addition to part geometry and tool motion statements, the programmer must also provide other instructions to operate the machine tool properly.



Sample workpart, like other parts, can be defined in terms of basic geometric elements, such as points, lines, and circles.

The computer's job

The computer's job in computer-assisted part programming consists of the following steps:

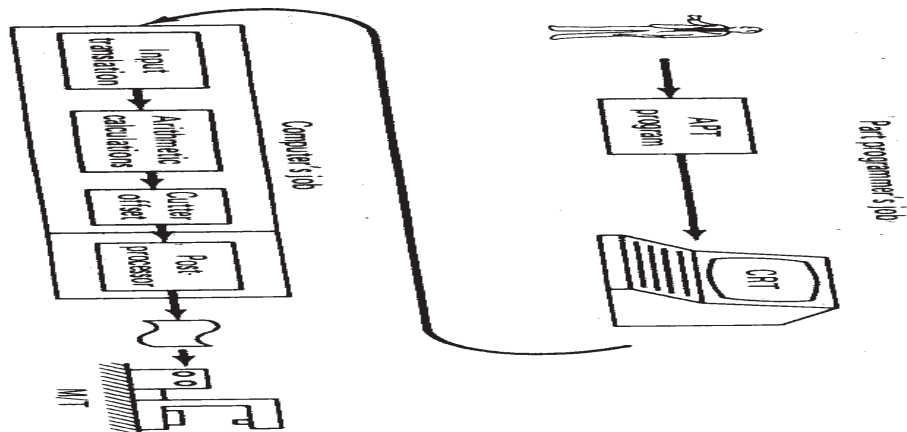
1. Input translation
2. Arithmetic calculations
3. Cutter offset computation
4. Postprocessor

The sequence of these steps and their relationships to the part programmer and the machine tool are illustrated in Figure.

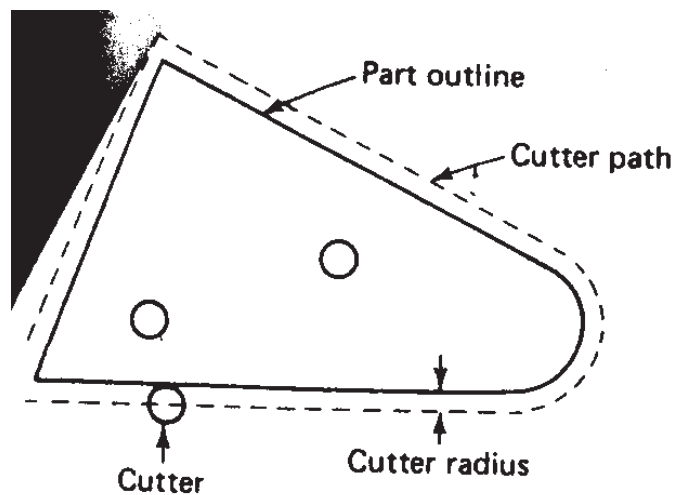
The part programmer enters the program written in the APT or other language. The input translation component converts the coded instructions contained in the program into computer-usable form, preparatory to further processing.

The arithmetic calculations unit of the system consists of a comprehensive set of subroutines for solving the mathematics required to generate the part surface. These subroutines are called by the various part programming language statements. The arithmetic unit is really the fundamental element in the part programming package. This unit frees the programmer from the time-consuming geometry and

trigonometry calculations, to concentrate on the workpart processing.



Steps in computer-assisted part programming.



Cutter offset problem in part programming for contouring.

The second task of the part programmer is that of constructing the tool path. However, the actual tool path is different from the part outline because the tool path is defined as the path taken by the center of the cutter. It is at the periphery of the cutter that machining takes place. The purpose of the cutter offset computations is to offset the tool path from the desired part surface by the radius of the cutter. This means that

the part programmer can define the exact part outline in the geometry statements. Thanks to the cutter offset calculation provided by the programming system, the programmer need not be concerned with this task. The cutter offset problem is illustrated in Figure.

As noted previously, NC machine tool systems are different. They have different features and capabilities. They use different NC tape formats. Nearly all of part programming languages, including APT, are designed to be general purpose languages, not limited to one or two machine tool types. Therefore, the main task of the computer in computer-assisted part programming is to take the general instructions and make them specific to a particular machine tool system. The unit that performs this task is called a postprocessor.

The postprocessor is a separate computer program that has been written to prepare the punched tape for a specific machine tool. The input to the postprocessor is output from the other three components: a series of cutter locations and other instructions. The output of the postprocessor is the NC tape written in the correct format for the machine on which it is to be used.

Part programming languages

NC part programming language consists of a software package (computer program) plus the special rules, conventions, and vocabulary words for using that software. Its purpose is to make it convenient for a part programmer to communicate the necessary part geometry and tool motion information to the computer so the desired part program can be prepared. The vocabulary words are typically English-like, to make the NC language easy to use.

There have probably been over 100 NC part programming languages developed since the initial MIT research on NC programming in the mid-1950s. Of the languages were developed to meet particular needs and have not survived the test of time. Today, there are several dozen NC languages still in use. Refinements and enhancements to existing languages are continually being made. The following list provides a description of some of the important NC languages in current use.

APT (Automatically Programmed Tools). The APT language was the

product of the MIT developmental work on NC programming systems. Its development began in June, 1956, and it was first used in production around 1959. Today it is the most widely used language in the United States. Although first intended as a contouring language, modern versions of APT can be used for both positioning and continuous-path programming in up to five axes. Versions of APT for particular processes include APTURN (for lathe operations), APTMIL (for milling and drilling operations), and APTPOINT (for point-to-point operations).

ADAPT (Adaptation of APT). Several part programming languages are based directly on the APT program. One of these is ADAPT, which was developed by IBM under Air Force contract. It was intended to provide many of the features of APT but to utilize a smaller computer. The full APT program requires a computing system that would have been considered large by the standards of the 1960s. This precluded its use by many small and medium-sized firms that did not have access to a large computer. ADAPT is not as powerful as APT, but it can be used to program for both positioning and contouring jobs.

EXAPT (Extended subset of APT). This was developed in Germany starting around 1964 and is based on the APT language. There are three versions:

EXAPT I—designed for positioning (drilling and also straight-cut milling), EXAPT II—designed for turning, and EXAPT III—designed for limited contouring operations. One of the important features of EXAPT is that it attempts to compute optimum feeds and speeds automatically.

UNIAPT. The UNIAPT package represents another attempt to adapt the APT language to use on smaller computers. The name derives from the developer, the United Computing Corp. of Carson, California. Their efforts have provided a limited version of APT to be implemented on minicomputers, thus allowing many smaller shops to possess computer-assisted programming capacity.

SPLIT (Sundstrand Processing Language Internally Translated). This is a proprietary system intended for Sundstrand's machine tools. It can handle up to five-axis positioning and possesses contouring capability as well. One of the unusual features of SPLIT is that the postprocessor is built into the program. Each machine tool uses its own SPLIT package, thus obviating the need for a special postprocessor.

COMPACT II. This is a package available from Manufacturing Data Systems, Inc. (MDSI), a firm based in Ann Arbor, Michigan. The NC language is similar to SPLIT in many of its features. MDSI leases the COMPACT II system to its users on a time-sharing basis. The part programmer uses a remote terminal to feed the program into one of the MDSI computers, which in turn produces the NC tape. The COMPACT II language is one of the most widely used programming languages. MDSI has roughly 3000 client companies which use this system.

PROMPT. This is an interactive part programming language offered by Weber N/C System, Inc., of Milwaukee, Wisconsin. It is designed for use with a variety of machine tools, including lathes, machining centers, flame cutters, and punch presses.

CINTURN II. This is a high-level language developed by Cincinnati Milacron to facilitate programming of turning operations.

The most widely used NC part programming language is APT, including its derivatives (ADAPT, EXAPT, UNIAPT, etc.).

THE APT LANGUAGE

In this section we present an introduction to the APT language for computer assisted part programming. Our objectives are to demonstrate the English-like statements of this NC language and to show how they are used to command the cutting tool through its sequence of machining operations.

APT is not only an NC language; it is also the computer program that performs the calculations to generate cutter positions based on APT statements. We will not consider the internal workings of the computer program. Instead, we will concentrate on the language that the part programmer must use.

APT is a three-dimensional system that can be used to control up to five axes. We will limit our discussion to the more familiar three axes, x, y, and z, and exclude rotational coordinates. APT can be used to control a variety of different machining operations. We will cover only drilling and milling applications. There are over 400 words in the APT vocabulary. Only a small (but important) fraction will be covered here.

There are four types of statements in the APT language:

1. *Geometry statements.* These define the geometric elements that comprise the workpart. They are also sometimes called definition statements.
2. *Motion statements.* These are used to describe the path taken by the cutting tool.
3. *Postprocessor statements.* These apply to the specific machine tool and control system. They are used to specify feeds and speeds and to actuate other features of the machine.
4. *Auxiliary statements.* These are miscellaneous statements used to identify the part, tool, tolerances, and so on.

Geometry statements

To program in APT, the workpart geometry must first be defined. The tool is subsequently directed to move to the various point locations and along surfaces of the workpart which have been defined by these geometry statements. The definition of the workpart elements must precede the motion statements.

The general form of an APT geometry statement is this:

An example of such a statement is

P1 = POINT/5.0, 4.0, 0.0

This statement is made up of three sections. The first is the symbol used to identify the geometric element. A symbol can be any combination of six or fewer alphabetic and numeric characters. At least one of the six must be an alphabetic character. Also, although it may seem obvious, the symbol cannot be one of the APT vocabulary words.

The second section of the geometry statement is an APT vocabulary word that identifies the type of geometry element. Besides POINT, other geometry elements in the APT vocabulary include LINE, PLANE and CIRCLE.

The third section of the geometry statement comprises the descriptive data that define the element precisely, completely, and uniquely. These data may include quantitative dimensional and positional data, previously defined geometry elements,

and other APT words.

The punctuation used in the APT geometry statement is illustrated in the example, Eq. .The statement is written as an equation, the symbol being equated to the surface type. A slash separates the surface type from the descriptive data. Commas are used to separate the words and-numbers in the descriptive data.

There are a variety of ways to specify the different geometry elements. The appendix at the end of this chapter presents a dictionary of APT vocabulary words as well as a sampling of statements for defining the geometry elements we will be using: points, lines, circles, and planes. The reader may benefit from a few examples.

To specify a line, the easiest method is by two points through which the line passes:

$$L3 = \text{LINE}/P3, P4$$

The part programmer may find it convenient to define a new line parallel to another line which has previously been defined. One way of doing this would be

$$L4 = \text{LINE}/P5, \text{PARLEL}, L3$$

This states that the line L4 must pass through point P5 and be parallel (PARLEL) line L3.

A plane can be defined by specifying three points through which it passes:

$$PL1 = \text{PLANE}/P1, P4, P5$$

Of course, the three points must not lie along a straight line. A plane can also be defined as being parallel to another plane, similar to the previous line parallelism statement.

$$PL2 = \text{PLANE}/P2, \text{PARLEL}, PL1$$

Plane PL2 is parallel to plane PL1 and passes through point P2.

A circle can be specified by its center and its radius.

$$C1 = \text{CIRCLF}/\text{CENTER}, P1, \text{RADIUS}, 5.0$$

The two APT descriptive words are used to identify the center and radius. The orientation of the circle perhaps seems undefined. By convention, it is a circle located in the x-y plane.

There are several ground rules that must be followed in formulating an APT geometry statement:

1. The coordinate data must be specified in the order x, y, z. For example, the statement

P1 = POINT/5.0, 4.0, 0.0

is interpreted by the APT program to mean a point x = 5.0, y = 4.0, and z = 0.0.

2. Any symbols used as descriptive data must have been previously defined. For example, in the statement

P2 = POINT/INTOF, L1, L2

the two lines L1 and L2 must have been previously defined. In setting up the list of geometry statements, the APT programmer must be sure to define symbols before using them in subsequent statements.

3. A symbol can be used to define only one geometry element. The same symbol cannot be used to define two different elements. For example, the following sequence would be incorrect:

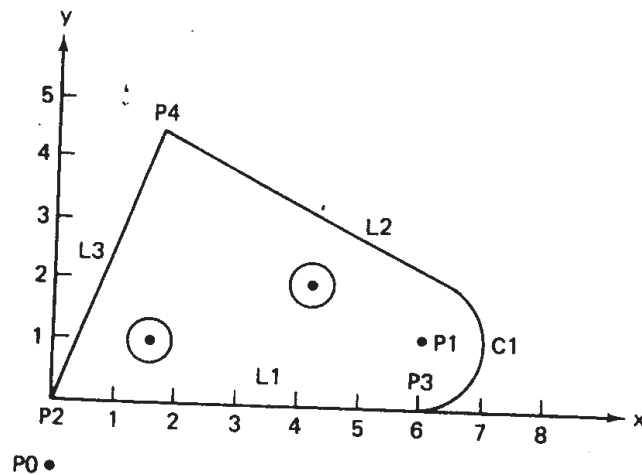
P1 = POINT/1.0, 1.0, 1.0

P1 = POINT/2.0, 3.0, 4.0

4. Only one symbol can be used to define any given element. For example, the following two statements in the same program would render the program incorrect:

P1 = POINT/1.0, 1.0, 1.0 n = POINT/1.0, 1.0, 1.0

5. Lines defined in APT are considered to be of infinite length in both directions. Similarly, planes extend indefinitely and circles defined in APT are complete circles.



Workpart from previous figure redrawn with x-y coordinate system and geometric elements labeled.

EXAMPLE

To illustrate some of these geometry statements we will define the geometry of the workpart shown in Figure 8.6. The drawing of the part is duplicated in Figure 8.9, except that we have added the coordinate axis system and labeled the various geometric elements. We also add the target point P0 to be used in subsequent motion commands.

```
P0 = POINT/0, -1.0, 0
P1 = POINT/6.0, 1.125, 0
P2 = POINT/0, 0, 0
P3 = POINT/6.0, 0, 0
P4 = POINT/1.75, 4.5, 0
L1 = LINE/P2, P3
C1 = CIRCLE/CENTER, P1, RADIUS, 1.125
L2 = LINE/P4, LEFT, TANTO, C1
L3 = LINE/P2, P4
PL1 = PLANE/P2, P3, P4
```

Motion statements

APT motion statements have a general format, just as the geometry statements do. The general form of a motion statement is

motion command/descriptive data

The example of a motion statement is

GOTO/PI

The statement consists of two sections separated by a slash. The first section is the basic motion command, which tells the tool what to do. The second section is comprised of descriptive data, which tell the tool where to go. In the example statement above, the tool is commanded to go to point PI, which has been defined in a preceding geometry statement.

At the beginning of the motion statements, the tool must be given a starting point. This point is likely to be the target point, the location where the operator has positioned the tool at the start of the job. The part programmer keys into this starting position with the following statement:

FROM/TARG

The FROM is an APT vocabulary word which indicates that this is the initial point from which others will be referenced. In the statement above, TARG is the symbol given to the starting point. Any other APT symbol could be used to define the target point. Another way to make this statement is

FROM/-2.0, -2.0, 0.0

where the descriptive data in this case are the x, y, and z coordinates of the target point. The FROM statement occurs only at the start of the motion sequence.

It is convenient to distinguish between PTP movements and contouring movements when discussing the APT motion statements.

POINT-TO-POINT MOTIONS. There are only two basic PTP motion commands: GOTO and GODLTA. The GOTO statement instructs the tool to go to a particular point location specified in the descriptive data. Two examples would be:

GOTO/P2

GOTO/2.0, 7.0, 0.0

In the first statement, P2 is the destination of the tool point. In the second statement, the tool has been instructed to go to the location whose coordinates are $x = 2.0$, $y = 7.0$, and $z = 0$.

The GODLTA command specifies an incremental move for the tool. For

example, the statement

```
GODLA/2.0, 7.0, 0.0
```

instructs the tool to move from its present position 2 in. in the x direction and 7 in. in the y direction. The z coordinate remains unchanged.

The GODLTA command is useful in drilling and related operations. The tool can be directed to a particular hole location with the GOTO statement. Then the GODLTA command would be used to drill the hole, as in the following sequence:

```
GOTO/P2
```

```
GODLTNO, 0, -1.5
```

```
GODLTNO, 0, + 1.5
```

EXAMPLE

Previous example was a PTP job which was programmed manually. Let us write the APT geometry and motion statements necessary to perform the drilling portion of this job. We will set the plane defined by $z = 0$ about 1/4 in. above the part surface. The part will be assumed to be 1/2 in. thick. The reader should refer back to Figures 8.3 and 8.4.

```
P1 = POINT/1.0, 2.0, 0
```

```
P2 = POINT/1.0, 1.0, 0
```

```
P3 = POINT/3.5, 1.5, 0
```

```
P0 = POINT/-1.0, 3.0, 2.0
```

```
FROM/P0
```

```
GOTO/P1
```

```
GODLTA/0, 0, -1.0
```

```
GODLTA/0, 0, + 1.0
```

```
GOTO/P2
```

```
GODLTA/0, 0, -1.0
```

```
GODLTA/0, 0, +1.0
```

GOTO/P3

GODLTA/0, 0, -1.0

GODLTA/0, 0, +1.0

GOTO/P0

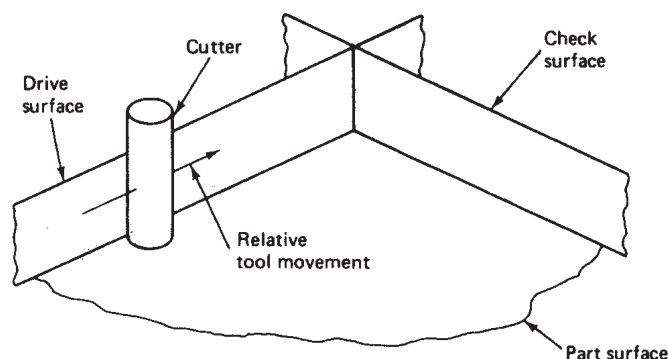
This is not a complete APT program because it does not contain the necessary auxiliary and postprocessor statements. However, the statement sequence demonstrates how geometry and motion statements can be combined to command the tool through a series of machining steps.

CONTOURING MOTIONS. Contouring commands are somewhat more complicated because the tool's position must be continuously controlled throughout the move. To accomplish this control, the tool is directed along two intersecting surfaces as shown in Figure 8.10. These surfaces have very specific names in APT:

1. *Drive surface.* This is the surface (it is pictured as a plane in Figure 8.10) that guides the side of the cutter.

2. *Part surface.* This is the surface (again shown as a plane in the figure) on which the bottom of the cutter rides. The reader should note that the "part surface" may or may not be an actual surface of the workpart. The part programmer must define this plus the drive surface for the purpose of maintaining continuous path control of the tool.

There is one additional surface that must be defined for APT contouring motions:



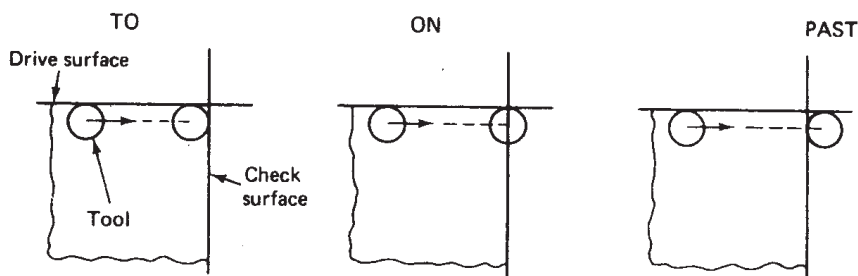
Three surfaces in APT contouring motions which guide the cutting tool.

3. *Check surface*. This is the surface that stops the movement of the tool in its current direction. In a sense, it checks the forward movement of the tool.

There are several ways in which the check surface can be used. This is determined by APT modifier words within the descriptive data of the motion statement. The three main modifier words are TO, ON, and PAST, and their use with regard to the check surface is shown in Figure. A fourth modifier word is TANTO. This is used when the drive surface is tangent to a circular check surface, as illustrated in Figure. In this case the cutter can be brought- to the point of tangency with the circle by use of the TANTO modifier word.

The APT contour motion statement commands the cutter to move along the drive and part surfaces and the movement ends when the tool is at the check surface. There are six motion command words:

GOLFT GOFWD GOUP
GORGT GOBACK GODOWN

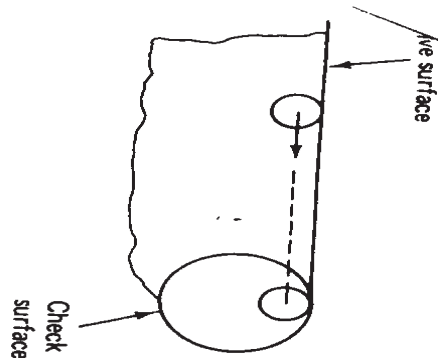


Use of APT modifier words in a motion statement: TO. ON and PAST. TO moves tool into initial contact with check surface. ON moves tool until tool center is on check surface. PAST moves tool just beyond check surface.

Previous figure Use of APT modifier word TANTO. TANTO moves tool to point of tangency between two surfaces, at least one of which is circular.

Their interpretation is illustrated in Figure. In commanding the cutter, the programmer must keep in mind where it is coming from. As the tool reaches the new check surface, does the next movement involve a right turn or an upward turn or what? The tool is directed accordingly by one of the six motion words. In the use of

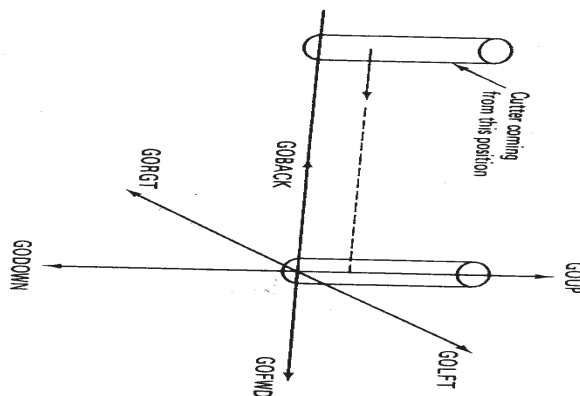
these words, it is helpful for the programmer to assume the viewpoint that the workpiece remains stationary and the tool is instructed to move relative to the piece. To begin the sequence of motion commands, the FROM statement, Eq. is used in the same manner , as for PTP moves. The statement following the FROM statement defines the initial drive surface, part surface, and check surface. The sequence is of the following form:



FROM/TARG

GO/TO, PL1, TO, PL2, TO, PL3

The symbol TARG represents the target point where the operator has set up the tool. The GO command instructs the tool to move to the intersection of the drive.



Use of APT motion commands.

surface (PL1), the part surface (PL2), and the check surface (PL3). The periphery of the cutter is tangent to PU and PU, and the bottom of the cutter is

touching P12. This cutter location is defined by use of the modifier word TO. The three surfaces included in the GO statement must be specified in the order: drive surface first, part surface second, and check surface last.

Note that the GO/TO command is different from the GOTO command.

GOTO is used only for PTP motions. GO/TO is used to initialize the sequence of contouring motions.

After initialization, the tool is directed along its path by one of the six command words. It is not necessary to repeat the symbol of the part surface after it has been defined. For instance, consider Figure. The cutter has been directed from TARG to the intersection of surfaces PU, PL2, and PU. It is desired to move the tool along plane PL3. The following command would be used:

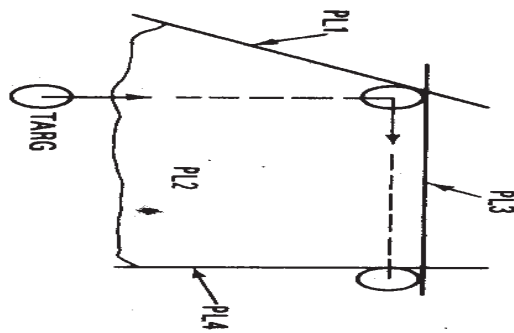
GORG/PL3, PAST, PL4

This would direct the tool to move along PL3, using it as the drive surface. The tool would continue until past surface PL4, which is the new check surface. Although the part surface (PL2) may remain the same throughout the motion sequence, the drive surface and check surface are redefined in each new command.

Let us consider an alternative statement to the above which would accomplish the same motion but would lead to easier programming:

GORG/U, PAST, L4

We have substituted lines L3 and L4 for planes PL3 and PL4, respectively. When looking at a part drawing, such as Figure, the sides of the part appear as lines. On the actual part, they are three-dimensional surfaces, of course. However, it would be more convenient for the part programmer to define these surfaces as lines and circles rather than planes and cylinders. Fortunately, the APT computer program allows the geometry of the part to be defined in this way. Hence the lines L3 and L4 in the foregoing motion statement are treated as the drive surface and check surface. This substitution can only be made when the part surfaces are perpendicular to the x-y plane.



Initialization of APT contouring motion sequence.

EXAMPLE

To demonstrate the use of the motion commands in a contouring sequence, we will refer back to the workpart of Example. The geometry statements were listed in this example for the part shown in Figure . Using the geometric elements shown in this figure, following is the list of motion statements to machine around the periphery of the part. The sequence begins with tool located at the target point PO.

```

FROM/PO

GO/TO, L1, TO, PL1, TO, L3

GORGTL/L1, TANTO, C1

GOFWD/C1, PAST, L2

GOFWD/L2, PAST, L3

GOLFT/L3, PAST, L1

GOTO/P0

```

The reader may have questioned the location of the part surface (PL1) in the APT sequence. For this machining job, the part surface must be defined below the bottom plane of the workpiece in order for the cutter to machine the entire thickness of the piece. Therefore, the part surface is really not a surface of the part at all.

Example raises several other questions: How is the cutter size accounted for in the APT program? How are feeds and speeds specified? These and other questions are answered by the postprocessor and auxiliary statements.

Postprocessor statements

To write a complete part program, statements must be written that control the operation of the spindle, the feed, and other features of the machine tool. These are called postprocessor statements. Some of the common postprocessor statements that appear in the appendix at the end of the chapter are:

```
COOLNT/  RAPID  
  
END      SPINDL/  
  
FEDRAT/  TURRET/  
  
MACHIN/
```

The postprocessor statements, and the auxiliary statements in the following section, are of two forms: either with or without the slash (/). The statements with the slash are self-contained. No additional data are needed. The APT words with the slash require descriptive data after the slash. These descriptions are given for each word in the appendix.

The FEDRAT/ statement should be explained. FEDRAT stands for feed rate and the interpretation of feed differs for different machining operations. In a drilling operation the feed is in the direction of the drill bit axis. However, in an end milling operation, typical for NC, the feed would be in a direction perpendicular to the axis of the cutter.

Auxiliary statements

The complete APT program must also contain various other statements, called auxiliary statements. These are used for cutter size definition, part identification, and so on. The following APT words used in auxiliary statements are defined in the appendix to this chapter:

```
CLPRNT  INTOL/  
  
CUTTER  OUTTOL/  
  
FINI    PARTNO
```

The offset calculation of the tool path from the part outline is based on the

CUTTER/definition. For example, the statement

CUTTER/.500

would instruct the APT program that the cutter diameter is 0.500 in. Therefore, the tool path must be offset from the part outline by 0.250 in.

EXAMPLE

We are now in a position to write a complete APT program. The workpart of Example 8.4 will be used to illustrate the format of the APT program.

We will assume that the workpiece is a low-carbon steel plate, which has previously been cut out in the rough shape of the part outline. The tool is a Y2-in.-diameter end-milling cutter. Typical cutting conditions might be recommended as follows: cutting speed = 573 rpm and feed = 2.29in.lmin.

Figure 8.15 presents the program with correct character spacing identified at the top as if it were to be keypunched onto computer cards. Modern NC programming systems utilize a CRT terminal for program entry.

THE MACROSTATEMENT IN APT

In the preceding section we described the basic ingredients of the APT language. In the present section we describe a very powerful feature of APT, the MACRO statement. The MACRO feature is similar to a subroutine in FORTRAN and other computer programming languages. It would be used where certain motion sequences would be repeated several times within a program. The purpose in using a MACRO subroutine is to reduce the total number of statements required in the APT program, thus making the job of the part programmer easier and less time

29

Column				72
1	6	8	10	
PARTNO			EXAMPLE PART	
			MACHIN/MILL, 1	
			CLPRNT	
			INTOL/.001	
			OUTTOL/.001	
			CUTTER/.500	
P0	=		POINT/0, -1.0, 0	
P1	=		POINT/6.0, 1.125, 0	
P2	=		POINT/0, 0, 0	
P3	=		POINT/6.0, 0, 0	
P4	=		POINT/1.75, 4.5, 0	
L1	=		LINE/P2, P3	
C1	=		CIRCLE/CENTER, P1, RADIUS, 1.125	
L2	=		LINE/P4, LEFT TANTO, C1	
L3	=		LINE/P2, P4	
PL1	=		PLANE/P2, P3, P4	
			SPINDL/573	
			FEDRAT/2.29	
			COOLNT/ON	
			FROM/P0	
			GO/TO, L1, TO, PL1, TO, L3	
			GORG/L1, TANTO, C1	
			GOFWD/C1, PAST, L2	
			GOFWD/L2, PAST, L3	
			GOLFT/L3, PAST, L1	
			RAPID	
			GOTO/P0	
			COOLNT/QFF	
			FINI	

APT program for Example

consuming. The MACRO subroutine is defined by a statement of the following format:

symbol = MACRO/parameter definition(s)

The rules for naming the MACRO symbol are the same as for any other APT symbol. It must be six characters or fewer and at least one of the characters must be a letter of the alphabet. The parameter definition(s) following the slash would identify certain variables in the subroutines which might change each time the subroutine was called into use. Equation (8.6) would serve as the title and first line of a MACRO subroutine. It would be followed by the set of APT statements that comprise the subroutine. The very last statement in the set must be the APT word TERMAC. This signifies the termination of the MACRO.

To activate the MACRO subroutine within an APT program, the following call statement would be used:

CALL/symbol, parameter specification

The symbol would be the name of the MACRO that is to be called. The parameter specification identifies the particular values of the parameters that are to be used in this execution of the MACRO subroutines.

An example will serve to illustrate the use of the MACRO statement and how the MACRO would be called by the main APT program.

EXAMPLE

We will refer back to the drilling operations of Example. In this example the GODLTA sequence was repeated in the program a total of three times, once for each hole. This represents an opportunity to use the MACRO feature in the APT system. The four point locations (P0, P1, and P3) would be defined just as they are in Example. These points would be used in the MACRO subroutine and main APT program in the following way:

```
DRILL = MACRO/PX  
  
GOTO/PX  
  
GODLTA/0, 0, -1.0  
  
GODLTA/0, 0, + 1.0  
  
TERMAC  
  
FROM/P0  
  
CALL/DRILL, PX = P1  
  
CALL/DRILL, PX = P2  
  
CALL/DRILL, PX = P3  
  
GOTO/PO
```

In this example the number of APT motion statements in the main program has been reduced from 11 down to five. (If we include the MACRO subroutine in our line count, the reduction is from 11 statements to 10.) The reader can visualize the power of the MACRO feature for a programming job in which there are a large number of holes to be drilled. The savings in the required number of APT statements

would approach $66^{2/3}\%$ in this case, since one call statement replaces three motion statements in the program.

The MACRO feature has many uses in APT. They are limited primarily by the imagination of the part programmer. Some of these uses will be considered in the exercise problems at the end of the chapter. It is even possible to have a CALL/ statement within one MACRO which refers to another MACRO subroutine. This might be used for example in a matrix of holes where both the x and y positions of the holes are changed with each drilling operation.