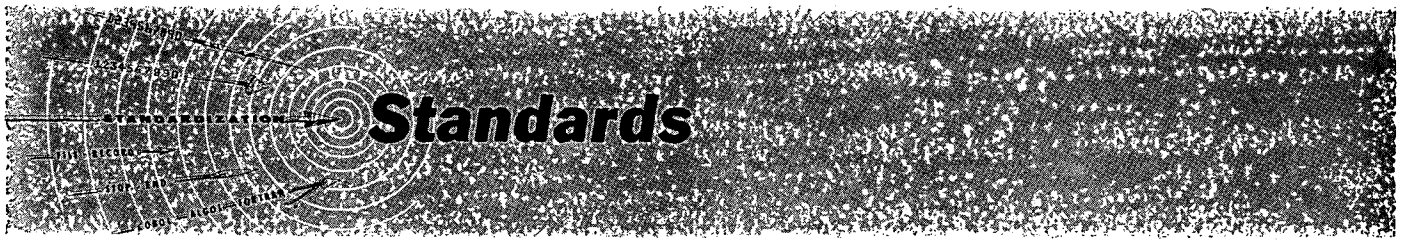


A Description of the APT Language*

S. A. BROWN, C. E. DRAYTON, B. MITTMAN
IIT Research Institute,† Chicago, Illinois

The APT (Automatically Programmed Tools) language for numerical control programming is described using the metalinguistic notation introduced in the ALGOL 60 report. Examples of APT usage are included. Presented also are an historical summary of the development of APT and a statement concerning its present status.

INTRODUCTION

The application of numerical control (N/C) to manufacturing has increased steadily in importance since its feasibility was demonstrated by MIT's Servo-mechanisms Laboratory in 1952. The introduction of the computer to assist in the preparation of the numerical control information has been a key to practical utilization of N/C for a variety of manufacturing processes. In contour milling, for example, it is often necessary to approximate a space curve by straight line segments (cuts) within a few thousandths of an inch tolerance. To accomplish this one may require the generation of thousands of coordinate points lying on the curve (or within tolerance of the curve) and for each of these points a cutter-radial offset must be calculated to determine the cutter center path.

A great number of computer systems have been developed for numerical control programming. Among these are APT, WALDO, AUTOPROMT, SPLIT, AUTOSPOT, AUTOMAP, etc. APT (Automatically Programmed Tools) is the most general and most widely used of these systems. The current APT system is presently available on one large scale computer and is being implemented on others. The APT system is in daily use in a number of manufacturing organizations.

In this paper the development of APT is summarized, its present status is discussed, and its language characteristics are described.

Historical Summary

In 1955, a prototype APT system was coded for the Whirlwind computer at MIT to demonstrate feasibility. This rudimentary version required the programmer to specify the endpoints of each straight line cut to be performed by the machine tool. In 1957, member organizations of the Aerospace Industries Association (AIA), in cooperation with MIT, undertook further development of the APT system. As a result of this development, a more advanced system was prepared for the IBM 704 in 1958. This system, called APT II, relieved the programmer of the responsibility of computing successive cutter locations and enabled him to describe the curve in an artificial language resembling English [1]. The APT system provided a language translator. This was the beginning of the APT language as we know it today, a language which permits the so-called part programmer to describe the part to be machined and the functions to be performed on that part.

Several versions of APT II have been used successfully in production by many aerospace companies. A still more effective system, known as APT III, was produced for the IBM 7090 as a cooperative AIA project and was released in December of 1961 [2].

During 1961, realizing that the APT concept was practical, but that its capabilities and potential had been just barely tapped, the AIA established the APT Long Range Program. Armour Research Foundation (now the IIT Research Institute) was selected to assume maintenance and validation responsibility for the existing APT system, and to direct the future course of a long-range developmental program. At the same time, the APT system, which previously had been available only to AIA members, was made available to any American company or government facility which desired to participate in the program. Companies joining the APT Long Range Program receive the complete APT system, documentation, training, etc., and by participating, help to underwrite the cost of further development.

* Received August 15, 1963.

† Formerly Armour Research Foundation of Illinois Institute of Technology.

Status of the APT System

Besides the APT system on the IBM 7090, Univac is developing APT for the UNIVAC 1107 and other computer implementations are being studied. In addition, under an Air Force contract early in 1963, IBM demonstrated the feasibility of implementing a substantial subset of APT on a small computer [3]. This subset, known as ADAPT, or comparable subsets are expected to be made available soon by a number of manufacturers of small computers.

Now that APT and its subsets will be gaining broadening utility, a group, initiated by the X3.4 Committee on Common Programming Languages of the American Standards Association, has begun to consider the desirability and feasibility of an organized standardization activity for APT. This study is to be conducted by a group of APT and ADAPT implementors and users under the auspices of the ASA subcommittee X3.4.2.

THE APT LANGUAGE

General Description

The APT language provides the same flexibility of expression to part programmers that standard programming languages provide to computer programmers. With APT the part programmer can define tool shapes, tolerances, geometric definitions, direction of motion of the tool, tool position relative to controlling surface, and auxiliary commands. In addition, the part programmer can write computational statements, macros, and looping statements. The present APT language has a vocabulary of approximately 275 words.

An APT *part program* consists of a sequence of *statements*, each of which contains at least one unit of information adequate in itself to activate one complete function of the APT system or to describe fully one pertinent condition or fact. Examples of APT statements are:

COOLNT/FLOOD	Turn on coolant at the flood setting
GO TO/SETPT	Move tool to a point symbolically designated as "SETPT"
L1 = LINE/PT1, PT2	L1 is the symbolic designation of a line passing through points PT1 and PT2

Most APT statements are divided into two sections separated by a slash. The "major" word appears to the left of the slash. The secondary section, if required, appears to the right of the slash and contains necessary modifiers to the major word. Nesting is allowable, i.e. secondary sections may themselves be sectionalized. For example:

GO TO/(POINT/LIN 1, LIN 2)	Move the tool to a point which is the intersection of two previously defined lines, LIN 1 and LIN 2.

The following sections contain a general description of the APT language as implemented for the IBM 7090. The notation used is identical to the metalanguage for syntactic description which was employed in previous ALGOL [4] and NELIAC [5] reports. The basic symbols of the metalanguage are:

- ::= connective meaning "is defined to be"
- | connective meaning "or"
- ⟨ ⟩ delimiting brackets enclosing metalinguistic variables

It should be pointed out that for brevity, the entire APT language has not been included in this report. An effort has been made, however, to present the syntax in sufficient detail to illustrate the scope of APT expression in numerical control programming.

1.0 Basic Symbols, Identifiers, Numbers, Strings

1.0.1 *Semantics.* The APT language is composed of a number of basic symbols.

1.0.2 *Syntax*

⟨basic symbol⟩ ::= ⟨letter⟩ | ⟨digit⟩ | ⟨delimiter⟩

1.1 LETTERS

1.1.1 *Semantics.* The upper-case, Roman alphabet is used to form identifiers and strings.

1.1.2 *Syntax*

⟨letter⟩ ::= A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

1.2 DIGITS

1.2.1 *Semantics.* Decimal digits are used to form identifiers, numbers, and strings.

1.2.2 *Syntax*

⟨digit⟩ ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

1.3 DELIMITERS

1.3.1 *Semantics.* Delimiters are combinations of one or more basic symbols which have fixed meanings within the language and therefore themselves form basic symbols.

1.3.2 *Syntax*

⟨delimiter⟩ ::= ⟨operator⟩ | ⟨separator⟩ | ⟨vocabulary word⟩

⟨separator⟩ ::= , | (|)

⟨operator⟩ ::= + | − | * | / | **

⟨vocabulary word⟩ ::= ⟨entry in APT vocabulary list⟩[1]

1.3.3 *Examples*

GOLFT
+
(
−
)
COOLNT
CUTTER

1.4 IDENTIFIERS

1.4.1 *Semantics.* Identifiers are strings of from one to six letters and digits. Except for a label which may consist entirely of digits, a valid identifier must include at least one letter.

1.4.2 *Syntax*

⟨identifier⟩ ::= ⟨letter⟩ | ⟨digit⟩ ⟨identifier⟩ | ⟨identifier⟩ ⟨digit⟩ | ⟨identifier⟩ ⟨letter⟩

[1] See Appendix A, APT Vocabulary List.

1.4.3 *Examples*

ABLE
123A6
74X
X74

1.5 NUMBERS

1.5.1 *Semantics.* A number consists of from 1 to 12 characters, including the decimal point. Considered as an integer, without a decimal point, a number may not be greater than 34,359,738,367 $= 2^{35} - 1$. Floating point is the only type of number used for APT computations.

1.5.2 *Syntax*

⟨number⟩ ::= ⟨unsigned number⟩ | + ⟨unsigned number⟩ | − ⟨unsigned number⟩
⟨unsigned number⟩ ::= ⟨unsigned integer⟩ | ⟨decimal fraction⟩ | ⟨unsigned integer⟩ ⟨decimal fraction⟩
⟨decimal fraction⟩ ::= .⟨unsigned integer⟩
⟨unsigned integer⟩ ::= ⟨digit⟩ | ⟨unsigned integer⟩ ⟨digit⟩

1.5.3 *Examples*

0
0.
0.1324
+7362
−7361
7360. 7360

1.6 STRINGS

1.6.1 *Semantics.* Strings are sequences of basic symbols not exceeding 66 characters in length.

1.6.2 *Syntax*

⟨string⟩ ::= ⟨any sequence of basic symbols⟩ | ⟨null⟩
⟨null⟩ ::=

1.6.3 *Examples*

29666XQLF**)()(
THIS IS A STRING

1.7 REMARKS

1.7.1 *Semantics.* Remarks are text inserted within the body of an APT program to enhance readability. They have no other significance.

1.7.2 *Syntax*

⟨remark⟩ ::= REMARK ⟨string⟩

1.7.3 *Example*

REMARK THIS IS A REMARK

2.0 Expressions

2.0.1 *Semantics.* An expression is a rule for defining a geometric entity or computing a numerical value.

2.0.2 *Syntax*

⟨expression⟩ ::= ⟨arithmetic expression⟩ | ⟨geometric expression⟩

2.1 VARIABLES

2.1.1 *Semantics.* A variable is an identifier that has been assigned as the name of the geometric entity or numerical value defined by an expression. It may be used in place of the expression at succeeding occurrences of that expression.

2.1.2 *Syntax*

⟨variable⟩ ::= ⟨simple variable⟩ | ⟨subscripted variable⟩
⟨subscripted variable⟩ ::= ⟨array identifier⟩ (⟨subscript expression⟩)
⟨array identifier⟩ ::= ⟨identifier⟩
⟨subscript expression⟩ ::= ⟨arithmetic expression⟩
⟨simple variable⟩ ::= ⟨identifier⟩
⟨arithmetic variable⟩ ::= ⟨variable assigned to an arithmetic expression⟩
⟨geometric variable⟩ ::= ⟨variable assigned to a geometric expression⟩

2.1.3 *Examples*

A
A(3)
A(A(6))

2.2 FUNCTION DESIGNATORS

2.2.1 *Semantics.* Function designators define single arithmetic values according to special algorithms built into the APT system.

2.2.2 *Syntax*

⟨function designator⟩ ::= ⟨function identifier⟩ (⟨function parameter list⟩)
⟨function parameter list⟩ ::= ⟨expression⟩ | ⟨variable⟩ | ⟨function parameter list⟩, ⟨function parameter list⟩
⟨function identifier⟩ ::= DOTF | SQRTF | SINF | COSF | EXPF | LOGF | ATANF | ABSF | LNTHF

2.2.3 *Examples*

SINF (A + B)
LNTHF (VECTOR/P1, P2)

2.3 ARITHMETIC EXPRESSIONS

2.3.1 *Semantics.* An arithmetic expression is a rule for computing a numerical value.

2.3.2 *Syntax*

⟨arithmetic expression⟩ ::= ⟨term⟩ | ⟨adding operator⟩ ⟨term⟩ | ⟨arithmetic expression⟩ ⟨adding operator⟩ ⟨term⟩
⟨term⟩ ::= ⟨factor⟩ | ⟨term⟩ ⟨multiplying operator⟩ ⟨factor⟩
⟨factor⟩ ::= ⟨primary⟩ | ⟨factor⟩ ** ⟨primary⟩
⟨primary⟩ ::= ⟨unsigned number⟩ | ⟨arithmetic variable⟩ | ⟨function designator⟩ | (⟨arithmetic expression⟩)
⟨multiplying operator⟩ ::= * | /
⟨adding operator⟩ ::= + | −

2.3.3 *Examples*

A + COSF(A+B+6.02/4)/JKL
A(7) + COSF(Q(6+I+X))

2.4 GEOMETRIC EXPRESSIONS

2.4.1 *Semantics.* A geometric expression is a rule for defining a geometric entity, such as circle, line, sphere, etc. The specific entity is specified by a geometric form word. For each geometric form there are from 1 to 14 different methods of definition. The specific organization of expressions, variables, and modifiers (a subclass of vocabulary words used as descriptors) in a parameter list determines which specific definition scheme is to be used. Several example geometric definitions are presented below.

2.4.2 *Syntax*

⟨geometric expression⟩ ::= ⟨geometric form⟩/⟨parameter list⟩

⟨geometric form⟩ ::= POINT | PLANE | CIRCLE | LINE | CYLNDR | ELLIPS | HYPERB | CONE | GCONIC | LCONIC | SPHERE | QADRIC | POLCON | TABCYL | MATRIX | VECTOR

⟨parameter list⟩ ::= (⟨expression⟩) | ⟨number⟩ | ⟨variable⟩ | ⟨parameter list⟩, ⟨modifier word⟩ | ⟨modifier word⟩, ⟨parameter list⟩ | ⟨parameter list⟩, ⟨parameter list⟩

⟨modifier word⟩ ::= ⟨vocabulary word⟩

2.4.3 *Examples*

POINT/3.5, 7, 9 defines a point at (3.5, 7, 9)

CIRCLE/CENTER, PT, RADIUS, 7 defines a circle with center at coordinates associated with variable PT and radius of 7.

CIRCLE/CENTER, (POINT/3.5, 7, 9), RADIUS, 7 defines a circle at (3.5, 7, 9) with a radius of 7.

CIRCLE/PT1, (POINT/3.5, 7, 9), PT4 defines a circle passing through coordinates of PT1, (3.5, 7, 9) and PT4.

POINT/CENTER, (CIRCLE/PT1, (POINT/3.5, 7, 9), PT4)) defines a point with coordinates at the center of the circle in the previous example.

3.0 Statements

3.0.1 *Semantics.* Statements are the basic operational and control units of the APT language.

3.0.2 *Syntax*

⟨statement⟩ ::= ⟨label⟩) ⟨unlabeled statement⟩ | ⟨unlabeled statement⟩

⟨unlabeled statement⟩ ::= ⟨assignment statement⟩ | ⟨cutter positioning statement⟩ | ⟨sequence control statement⟩ | ⟨post processor control statement⟩ | ⟨procedure statement⟩ | ⟨input-output control statement⟩

⟨loop⟩ ::= LOOPST ⟨statement list⟩ | LOOPND

⟨statement list⟩ ::= ⟨statement⟩ | ⟨statement list⟩ ⟨statement⟩

3.1 Assignment Statements

3.1.1 *Semantics.* Assignment statements assign variables to the values of arithmetic expressions and to the definitions of geometric expressions. Once a variable is assigned to a geometric expression it may never be reassigned. This is not true, however, for variables assigned to arithmetic expressions. They may be reassigned at will.

3.1.2 *Syntax*

⟨assignment statement⟩ ::= ⟨variable⟩ = ⟨arithmetic expression⟩ | ⟨variable⟩ = ⟨geometric expression⟩

3.1.3 *Examples*

A = 2 + 3 + 1

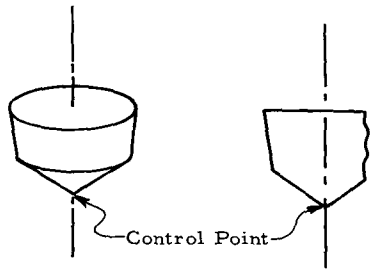
A = A + 1

B = CIRCLE/(POINT/0, 0, 0), PTA, PTB

3.2 Cutter Positioning Statements

3.2.1 *Semantics.* Numerically controlled machine tools are controlled by reference to a control point on the cutter.

Two general classes of commands are available to control positioning of this control point.



The first, explicit positioning control, specifies a new position in terms of actual coordinates or increments from the present position.

The next class provides for continuous cutter motion along a space curve. As the cutter, a surface of revolution, errodes material at its periphery, a useful construct, is to consider the cutter constrained by a tangency condition to each of two surfaces, the so-called part surface and drive surface. The position of the cutter relative to the two controlling surfaces is specified by continuous motion statements. These necessarily involve a direction of motion based on previous statements or declarations. Such a continuous motion is terminated by also specifying a tangency condition to a third surface called a check surface. When this required terminal tangency condition is met, the motion is complete.

3.2.2 *Syntax*

⟨cutter positioning statement⟩ ::= ⟨explicit positioning statement⟩ | ⟨initial continuous motion statement⟩ | ⟨intermediate continuous motion statement⟩ | ⟨terminal continuous motion statement⟩

3.3 Explicit Positioning Statements

3.3.1 *Semantics.* For those occasions when the desired location of the control point is known, the cutter can be directly positioned without the use of controlling surfaces by use of the explicit positioning statements.

GO TO — positions the control point at the specified coordinates.

GO DLTA — positions the control point in a specified increment from its current location.

FROM — indicates the initial position of the control point.

3.3.2 *Syntax*

⟨explicit positioning statement⟩ ::= GO TO/((geometric expression)) | GO TO/⟨geometric variable⟩ | GO TO/⟨arithmetic parameter⟩, ⟨arithmetic parameter⟩, ⟨arithmetic parameter⟩ | GO DLTA/⟨arithmetic parameter⟩, ⟨arithmetic parameter⟩, ⟨arithmetic parameter⟩ | FROM/((geometric expression)) | FROM/⟨geometric variable⟩ | FROM/⟨arithmetic parameter⟩, ⟨arithmetic parameter⟩, ⟨arithmetic parameter⟩

⟨arithmetic parameter⟩ ::= ((arithmetic expression)) | ⟨number⟩ | ⟨arithmetic variable⟩

3.3.3 *Examples*

GO TO/3.5, 4.7, .0037
GO TO/(POINT/2.1, 1.2, 0)
GO DLTA/0, .333, .6
FROM/1, 2, 3
FROM/PT4

3.4 Initial Continuous Motion Statements

3.4.1 *Semantics.* A special statement of this type is needed before each group of intermediate continuous motion statements in order to position the cutter within a specified tolerance from, and on the correct side of the part and drive surfaces. This statement also allows the cutter to be directed to a particular area of the controlling surfaces.

3.4.2 *Syntax*

⟨initial continuous motion statement⟩ ::= GO/⟨qualified geometric expression⟩ | GO/⟨qualified geometric expression⟩, ⟨qualified geometric expression⟩ | GO/⟨qualified geometric expression⟩, ⟨qualified geometric expression⟩, ⟨qualified geometric expression⟩ | OFFSET/⟨qualified geometric expression⟩

⟨qualified geometric expression⟩ ::= ⟨cutter tangency specifier⟩, ((geometric expression)) | ⟨cutter tangency specifier⟩, ⟨geometric variable⟩ | ((geometric expression)) | ⟨geometric variable⟩

⟨cutter tangency specifier⟩ ::= TO | ON | PAST | TANTO

3.4.3 *Examples*

GO/C3
GO/TO, C3
GO/PAST, C2, ON, LN7
GO/A, B, C
GO/ON, PLN5, PAST, LIN2, TO, CYL9

3.5 Intermediate Continuous Motion Statements

3.5.1 *Semantics.* The major use of the APT language is to direct a defined cutter to move in a specified direction, yet, maintain a specified tangency position relative to two geometric surfaces. The direction and position conventions are specified in Figure 1. Each statement is dependent upon the preceding statement for establishing the direction of motion and upon the drive surface of the next statement for information concerning when the motion is complete. The drive surface of this following statement is then also the check surface for the current motion.

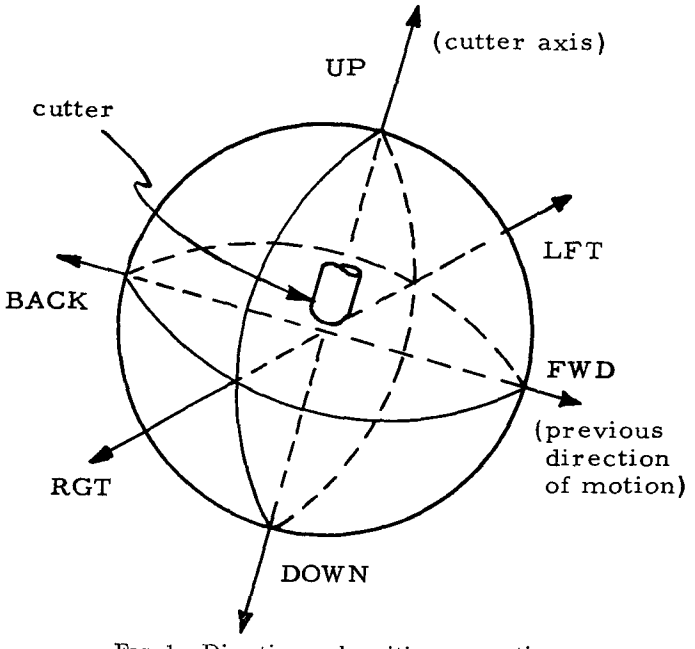


Fig. 1. Direction and position conventions

3.5.2 *Syntax*

⟨intermediate continuous motion statement⟩ ::= ⟨continuous motion word⟩/⟨drive surface⟩

⟨continuous motion word⟩ ::= GO LFT | GO RGT | GO FWD | GO BACK | GO UP | GO DOWN

⟨drive surface⟩ ::= ((geometric expression)) | ⟨geometric variable⟩

3.5.3 *Examples*

GO LFT/LN2
GO FWD/CIRL
GO RGT/(LINE/P1, P2)

3.6 Terminal Continuous Motion Statements

3.6.1 *Semantics.* The terminal continuous motion statement requires no successor statement from which to derive a check surface. It must, therefore, include reference to an explicit check surface and end tangency condition. It may occur from time to time that the geometric configuration is such that the cutter may satisfy the required check surface tangency criterion at more than one point. In this case the motion is considered complete at the first satisfaction of this constraint. A command format is available to allow terminating the motion at the n-th satisfaction of the tangency condition.

3.6.2 *Syntax*

⟨terminal continuous motion statement⟩ ::= ⟨intermediate continuous motion statement⟩, ⟨cutter tangency specifier⟩, ⟨check surface⟩ | ⟨intermediate continuous motion statement⟩, ⟨check surface⟩ | ⟨intermediate continuous motion statement⟩, ⟨cutter tangency specifier⟩, ⟨intersection count⟩, INTOF, ⟨check surface⟩

⟨check surface⟩ ::= ((geometric expression)) | ⟨geometric variable⟩

⟨intersection count⟩ ::= ⟨unsigned integer⟩

3.6.3 *Examples*

GO LFT/LN2, TO, CIR6
GO RGT/(LINE/P3, P6), CIRX
GO BACK/LX4361, ON, 4, INTOF, POL6

3.7 Sequence Control St tements

3.7.1 *Semantics.* A sequence control statement interrupts the normal sequential execution of statements.

3.7.2 *Syntax*

⟨sequence control statement⟩ ::= ⟨arithmetic transfer statement⟩ | ⟨geometric transfer statement⟩ | ⟨termination statement⟩

3.8 Arithmetic Transfer Statement

3.8.1 *Semantics.* An arithmetic transfer statement and the statements associated with labels appearing in these transfer statements must occur within a loop or procedure. If normal sequential execution is altered by an arithmetic transfer statement, it can only be re-established by another arithmetic transfer.

3.8.2 *Syntax*

⟨arithmetic transfer statement⟩ ::= JUMPTO/⟨label⟩ | IF (⟨arithmetic expression⟩) ⟨label⟩, ⟨label⟩, ⟨label⟩

3.8.3 *Examples*

```
JUMPTO/123XQ
IF (A-X) ST1, ST2, ST3
```

3.9 Geometric Transfer Statements

3.9.1 *Semantics.* A geometric transfer statement also interrupts sequential execution of an APT program. However, it need not appear within a loop or procedure. If a geometric transfer statement does appear in a loop or procedure it may only reference statements occurring later within the program. The same rule applies concerning consistent use of geometric transfers as applies to arithmetic transfers. A multiple choice surface motion statement selects one of two possible successor statements depending upon which tangency condition is first achieved.

3.9.2 *Syntax*

⟨geometric transfer statement⟩ ::= TRANTO/⟨label⟩ | ⟨multiple check surface motion statement⟩

⟨multiple check surface motion statement⟩ ::= ⟨terminal continuous motion statement⟩, ⟨label⟩, ⟨check surface⟩, ⟨label⟩ | ⟨terminal continuous motion statement⟩, ⟨label⟩, ⟨cutter tangency specifier⟩, ⟨check surface⟩, ⟨label⟩

3.9.3 *Example*

```
      TLRGT
      GO FWD/CIRC
      GO FWD/L1, TO, L2, ID1, TO, L3, ID2
ID1)  TLLFT, GO LFT/L2, PAST, L4
      TRANTO/ID3
ID2)  TLLFT, GO LFT/L3, TO, L2, GO LFT/L2,
        PAST, L4
ID3)  - - - - - - - - - -
```

3.10 Termination Statements

3.10.1 *Semantics.* This statement defines no successor and has the effect of terminating the program.

3.10.2 *Syntax*

⟨termination statement⟩ ::= FINI

3.11 Post Processor Control Statements

3.11.1 *Semantics.* A post processor in the APT system is a program which transforms general coordinate information and control functions into codes for a specific controller-machine tool configuration. Post processor control commands activate specific internal functions of a post-processor. These are to be differentiated from the strings of coordinates produced by cutter positioning statements. Postprocessor control statements control such internal functions as spindle speed and direction, servo mechanism overshoot, coolant values, etc.

3.11.2 *Syntax*

⟨postprocessor control statement⟩ ::= ⟨postprocessor control word⟩ | ⟨postprocessor control word⟩/⟨parameter list⟩ | ⟨postprocessor control word⟩ ⟨string⟩ | ⟨cutter positioning statement⟩, ⟨feedrate specifier⟩

⟨postprocessor control word⟩ ::= ⟨vocabulary word⟩

⟨feedrate specifier⟩ ::= (⟨arithmetic expression⟩) | ⟨arithmetic variable⟩ | ⟨number⟩

3.11.3 *Examples*

```
COOLNT/ON
SPINDL/ON
PPRINT ARBITRARY TEXT
END
```

3.12 Procedure Statements

3.12.1 *Semantics.* A procedure statement is replaced by the procedure body. The formal parameters within the body are replaced by the normal parameters specified in the procedure declaration except where superceded by actual parameters from the procedure statement. An APT procedure statement calls by name.

3.12.2 *Syntax*

⟨procedure statement⟩ ::= CALL/⟨procedure identifier⟩ | CALL/⟨procedure identifier⟩, ⟨procedure parameter list⟩

⟨procedure parameter list⟩ ::= ⟨formal parameter⟩ = ⟨actual parameter⟩ | ⟨procedure parameter list⟩, ⟨formal parameter⟩ = ⟨actual parameter⟩

⟨actual parameter⟩ ::= ⟨variable⟩ | ⟨number⟩ | ⟨vocabulary word⟩ | ⟨label⟩

⟨formal parameter⟩ ::= ⟨identifier⟩

3.12.3 *Examples*

```
CALL/MACR, A = GORGT, B = CIRCLE, C = QLX,
  D = 7.632
CALL/MXR
```

3.13 Input-Output Control Statements

3.13.1 *Semantics.* These statements allow for the input and output of procedures, geometric entities, and numerical quantities.

3.13.2 *Syntax*

⟨input-output control statement⟩ ::= ⟨input-output control word⟩/⟨format specifier⟩, ⟨I-O list⟩ | TITLES ⟨string⟩

⟨input-output control word⟩ ::= READ | PUNCH | PRINT

⟨I-O list⟩ ::= ALL | ⟨variable list⟩

⟨variable list⟩ ::= ⟨variable⟩ | ⟨variable list⟩, ⟨variable⟩

⟨format specifier⟩ ::= 0 | 1 | 2 | 3

3.13.3 *Examples*

```
PRINT/3, ALL
PUNCH/1, X, HQC, J
READ/1, A, B, X
TITLES SIN COS TAN
```

4.0 Declarations

4.0.1 *Semantics.* Declarations define properties of the geometric entities and arithmetic quantities used within a program. They also alter the environment in which certain statements are executed. An environment defined by a declaration remains in effect until it is superceded.

4.0.2 *Syntax*

⟨declaration⟩ ::= ⟨array declaration⟩ | ⟨coordinate transformation declaration⟩ | ⟨Z surface declaration⟩ | ⟨procedure declaration⟩ | ⟨vocabulary equivalence declaration⟩ | ⟨cutter offset calculation declarations⟩

4.1 Array Declarations

4.1.1 *Semantics.* An array declaration declares one or more identifiers to represent linear arrays of quantities or geometric entities.

4.1.2 *Syntax*

⟨array declaration⟩ ::= RESERV/⟨array list⟩
⟨array list⟩ ::= ⟨array segment⟩ | ⟨array list⟩, ⟨array segment⟩
⟨array segment⟩ ::= ⟨array identifier⟩, (⟨arithmetic expression⟩) | ⟨array identifier⟩, ⟨arithmetic variable⟩

4.1.3 *Examples*

RESERV/A, 12, B, 26, X, C

4.2 Coordinate Transformation Declarations

4.2.1 *Semantics.* This declaration creates a special environment in which geometric variables may be defined. When a variable so defined is referenced in the normal environment, it appears with its coordinates transformed as specified in the declaration controlling the definition environment. The word NO MORE re-instates the normal environment.

4.2.2 *Syntax*

⟨coordinate transformation declaration⟩ ::= REFSYS/⟨transformation matrix⟩ | REFSYS/NO MORE
⟨transformation matrix⟩ ::= ⟨geometric variable⟩

4.2.3 *Examples*

REFSYS/M
REFSYS/NO MORE

4.3 Z-Surface Declarations

4.3.1 *Semantics.* A point is one of the geometric entities that may be defined in an APT program. At the time of definition, unless explicitly defined otherwise, the point is assumed to lie on a plane containing the x and y axes. This declaration allows the x-y plane assumption to be overruled and declares points to here after lie on the specified plane.

4.3.2 *Syntax*

⟨Z surface declaration⟩ ::= ZSURF/((geometric expression)) | ZSURF/⟨geometric variable⟩

4.3.3 *Example*

ZSURF/PL2
ZSURF/(PLANE/PNT1, PNT2, PNT3)

4.4 Procedure Declarations

4.4.1 *Semantics.* A procedure declaration defines the procedure associated with a procedure identifier. When the procedure is referenced by a procedure statement, the identifiers within the procedure body declared to be formal parameters in the procedure heading will be replaced by the names of the corresponding actual parameters. At the termination of this process any formal parameter not replaced by an actual parameter will be replaced by the normal name corresponding to that formal parameter in the procedure heading.

4.4.2 *Syntax*

⟨procedure declaration⟩ ::= ⟨procedure heading⟩ ⟨procedure body⟩ ⟨procedure terminator⟩
⟨procedure heading⟩ ::= ⟨procedure identifier⟩ = MACRO/⟨formal parameter list⟩
⟨formal parameter list⟩ ::= ⟨null⟩ | ⟨formal parameter⟩ | ⟨formal parameter⟩ = ⟨normal name⟩ | ⟨formal parameter list⟩, ⟨formal parameter list⟩
⟨normal name⟩ ::= ⟨number⟩ | ⟨vocabulary word⟩ | ⟨label⟩
⟨procedure body⟩ ::= ⟨statement⟩ | ⟨procedure body⟩ ⟨statement⟩
⟨procedure terminator⟩ ::= TERMAC
⟨procedure identifier⟩ ::= ⟨identifier⟩

4.4.3 *Example*

```
MAC = MACRO/J = TLLFT, K, Z, H = 0
      FROM/0, 0, 0
      GO TO/1, 1, 1
      GO/SURF1
      J, GO LFT/SURF1
      GO RGT/SURF2, K, Z, H
      TERMAC
```

4.5 Vocabulary Equivalence Declarations

4.5.1 *Semantics.* This declaration allows an arbitrary identifier to be made equivalent to an APT vocabulary word.

4.5.2 *Syntax*

⟨vocabulary equivalence declaration⟩ ::= SYN/⟨equivalence list⟩
⟨equivalence list⟩ ::= ⟨identifier⟩, ⟨vocabulary word⟩ | ⟨equivalence list⟩, ⟨identifier⟩, ⟨vocabulary word⟩

4.5.3 *Example*

SYN/GT, GO TO, TT, TANTO

4.6 Cutter Offset Calculation Declarations

4.6.1 *Semantics.* These statements create an environment for cutter offset control statements.

4.6.2 *Syntax*

⟨cutter offset calculation declaration⟩ ::= ⟨direction declaration⟩ | ⟨calculation parameter declaration⟩ | ⟨tool position declaration⟩

4.7 Direction Declarations

4.7.1 *Semantics.* This declaration establishes or re-establishes the direction of tool motion. It is used primarily to resolve otherwise ambiguous initial continuous motion commands, such as when it is desired to move the cutter to a specified

position relative to a circle and the initial cutter location is, indeed, the very center of the circle. Two alternative declarations allow specifying the direction in terms of a vector or toward a point.

4.7.2 *Syntax*

⟨direction declaration⟩ ::= INDIRP/⟨direction specifier⟩ | INDIRV/⟨direction specifier⟩
⟨direction specifier⟩ ::= (⟨geometric expression⟩) | ⟨geometric variable⟩ | ⟨arithmetic parameter⟩, ⟨arithmetic parameter⟩, ⟨arithmetic parameter⟩

4.7.3 *Examples*

INDIRP/(POINT/1, 1, 1)
INDIRP/1, 1, 1
INDIRV/7, 6, 3
INDIRV/(VECTOR/P1, P2)
INDIRV/VECT1

4.8 Calculation Parameter Declarations

4.8.1 *Semantics.* These declarations specify parameters and arithmetic and logical constants for the internal calculations that reduce continuous motion statements to sequences of discrete cutter center coordinates.

4.8.2 *Syntax*

⟨calculation parameter declaration⟩ ::= ⟨tolerance specification⟩ | ⟨cutter specification⟩ | ⟨calculation constant control⟩

4.9 Tolerance Specifications

4.9.1 *Semantics.* All continuous motion commands are reduced to sequences of straight line motions departing from the analytic cutter tangency constraints by no more than a specified tolerance. Inner tolerance (INTOL) affects under-cutting or "gouge" while outer tolerance (OUTTOL) affects amount of excess stock.

4.9.2 *Syntax*

⟨tolerance specification⟩ ::= TOLER/⟨arithmetic parameter⟩ | INTOL/⟨arithmetic parameter⟩ | OUTTOL/⟨arithmetic parameter⟩

4.9.3 *Example*

TOLER/.0001
INTOL/(A/4.621)
OUTTOL/Q

4.10 Cutter Specifications

4.10.1 *Semantics.* The cutter, a surface of revolution, is defined by its profile after Figure 2.

4.10.2 *Syntax*

⟨cutter specification⟩ ::= CUTTER/⟨d⟩ CUTTER/⟨d⟩, | ⟨r⟩ CUTTER/⟨d⟩, ⟨r⟩, ⟨e⟩, ⟨f⟩, ⟨α⟩, ⟨β⟩, ⟨h⟩
⟨d⟩ ::= ⟨arithmetic parameter⟩
⟨r⟩ ::= ⟨arithmetic parameter⟩
⟨e⟩ ::= ⟨arithmetic parameter⟩
⟨f⟩ ::= ⟨arithmetic parameter⟩
⟨α⟩ ::= ⟨arithmetic parameter⟩
⟨β⟩ ::= ⟨arithmetic parameter⟩
⟨h⟩ ::= ⟨arithmetic parameter⟩

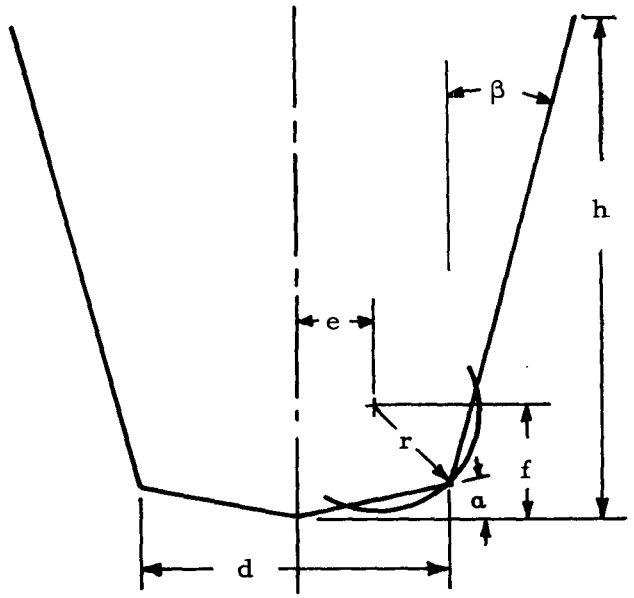


FIG. 2

4.10.3 *Example*

CUTTER/1.02
CUTTER/6, 4.3, 7.2, 9, C, (C+.07), A

4.11 Calculation Constant Controls

4.11.1 *Semantics.* Two broad categories of declarations control arithmetic and logic constants used in cutter offset calculations.

CUT and DNTCUT respectively, restore and inhibit forwarding of coordinates resulting from cutter positioning statements to the postprocessor.

MULTAX initiates output of tool axis direction cosines to the postprocessor.

TLAXIS specifies the direction cosines of tool axis. The NORMPS mode specifies that the tool axis is to be everywhere normal to the part surface.

MAXDP specifies the maximum step length to be allowed in cutter offset calculations.

NUMPTS specifies the maximum number of points that a single continuous motion command may produce.

THICK adds a uniform excess to cutter control surfaces.

PSIS specifies a new part surface.

4.11.2 *Syntax*

⟨calculation constant control⟩ ::= ⟨calculation logic control⟩ | ⟨calculation arithmetic constant control⟩
⟨calculation logic control⟩ ::= CUT | DNTCUT | MULTAX | 3DCALC | 2DCALC | NDTEST | NOPS
⟨calculation arithmetic constant control⟩ ::= TLAXIS/⟨arithmetic parameter⟩, ⟨arithmetic parameter⟩, ⟨arithmetic parameter⟩ | TLAXIS/(⟨geometric expression⟩) | TLAXIS/⟨geometric variable⟩ | TLAXIS/NORMPS | MAXDP/⟨arithmetic parameter⟩, NUMPTS/ ⟨arithmetic parameter⟩ | THICK/⟨arithmetic parameter⟩, ⟨arithmetic parameter⟩, ⟨arithmetic parameter⟩ | PSIS/(⟨geometric expression⟩) | PSIS/⟨geometric variable⟩

4.11.3 *Examples*

```
CUT
DNT CUT
TLAXIS/(VECTOR/PT1, PT2)
MAXDP/10
THICK/0, 0, .02
```

4.12 Tool Position Declarations

4.12.1 *Semantics.* The tool position declaration is used in lieu of a cutter tangency specifier in intermediate continuous motion statements. The following rule applies:

Tool Position	*Motion*	*Tangency Specifier*
TLLFT	GOLFT	TO
TLLFT	GORGT	PAST
TLRGT	GOLFT	PAST
TLRGT	GORGT	TO
TLON	GOLFT	ON
TLON	GORGT	ON
any	GOFWD	TANTO
any	GOBACK	TANTO
any	GOUP	error
any	GODOWN	error

For a tool position and motion specified in statement n, the tangency specifier refers to the check surface for statement $n - 1$.

4.12.2 *Syntax*

⟨tool position declaration⟩ ::= ⟨tool position specifier⟩ | ⟨tool position specifier⟩, ⟨continuous motion statement⟩ | ⟨tool position specifier⟩, ⟨tool position declaration⟩

⟨tool position specifier⟩ ::= TL ON PS | TL OF PS | TL ND ON | TANCRV | TLON | TLLFT | TLRGT

⟨continuous motion statement⟩ ::= ⟨intermediate continuous motion statement⟩ | ⟨terminal continuous motion statement⟩

REFERENCES

1. Ross, Douglas T. The design and use of the APT language for automatic programming of numerically controlled machine tools. Proc. 1959 Computer Applications Symposium, Chicago, pp. 80–99.
2. Bates, Edgar A. Automatic programming for numerically controlled tools—APT III. In *Computer Applications—1961*, pp. 140–156; Macmillan, New York.
3. International Business Machines Corporation. ADAPT, a system for the automatic programming of numerically controlled machine tools on small computers. Final Tech. Eng. Report, 15 July 1962-16 January 1963, San Jose, Calif. (Air Force Contract AF 33(600)-43365).
4. Naur, Peter, et al. Revised report on the algorithmic language ALGOL 60. *Comm. ACM 6* (Jan. 1963), 1–17.
5. Huskey, H. D., Love, R., and Wirth, N. A syntactic description of BC NELIAC. *Comm. ACM 6* (July 1963), 367–375.

APPENDIX A. APT VOCABULARY LIST

1.7 Remarks
REMARKS

2.2 Function Designators

DOTF	SINF	LOGF
LNTHF	COSF	ATANF
SQRTF	EXPF	ABSF

2.4 Geometric Expressions

POINT	ELLIPS	VECTOR
LINE	HYPERB	MATRIX
PLANE	CONE	SPHERE
CIRCLE	GCONIC	QADRIC
CYLNDR	LCONIC	POLCON
		TABCYL

3.0 Statements

LOOPST	LOOPND

3.3 Explicit Positioning Statements

FROM	GODLTA	GOTO

3.4 Initial Continuous Motion Statements

GO	OFFSET

3.5 Intermediate Continuous Motion Statements

GOLFT	GOFWD	GOUP
GORGT	GOBACK	GODOWN

3.8 Arithmetic Transfer Statements

IF	JUMPTO

3.9 Geometric Transfer Statements
TRANTO

3.10 Termination Statements
FINI

3.11 Post Processor Control Statement

END	DELAY	TRANS
STOP	AIR	TRACUT
OPSTOP	OPSKIP	INDEX
ISTOP	LEADER	COPY
RAPID	PPLOT	PREFUN
SWITCH	MACHIN	COUPLE
RETRCT	MCHTOL	PITCH
DRESS	PIVOTZ	CLAMP
PICKUP	MCHFIN	ENDMDI
UNLOAD	SEQNO	ASLOPE
PENUP	INTCOD	SADDLE
PENDWN	DISPLY	LOADTL
ZERO	AUXFUN	SELCTL
CODEL	CHECK	CLEARC
RESET	POSTN	CYCLE
PLABEL	TOOLNO	DRAFT
PLUNGE	ROTABL	RITMIDI
HEAD	ORIGIN	PLOT
MODE	SAFETY	OVPLOT
CLEARP	ARCSLP	LETTER
TMARK	COOLNT	PPRINT
REWIND	SPINDL	PARTNO
CUTCOM	TURRET	INSERT
REVERS	ROTHED	CAMERA
FEDRAT	THREAD	

3.12 Procedure Statements
CALL

3.13 Input-Output Control Statements

PRINT	READ	TITLES
PUNCH		

4.1 Array Declarations
RESERV

4.2 Coordinate Transformation Declarations
REFSYS

4.3 *Z*-Surface Declarations
ZSURF

4.4 Procedure Declarations

MACRO	TERMAC

4.5 Vocabulary Equivalence Declarations
SYN

4.6 Direction Declarations

INDIRP	INDIRV

4.9 Tolerance Specifications

TOLER	INTOL	OUTTOL

4.10 CUTTER SPECIFICATIONS

CUTTER

4.11 CALCULATION CONSTANT CONTROLS

CUT	NDTEST	NUMPTS
DNTCUT	TLAXIS	THICK
2DCALC	MULTAX	NOPS
3DCALC	MAXDP	PSIS

4.12 TOOL POSITION DECLARATIONS

TLLFT	TLNDON	TANCRV
TLRGT	TLON	TLONPS
		TLOFPS

MODIFIER WORDS

ATANGL	PARLEL	XYPLAN
CENTER	PERPTO	XYROT
CROSS	PLUS	YLARGE
FUNOFY	POSX	YSMALL
INTOF	POSY	YZPLAN
INVERS	POSZ	YZPLAN
LARGE	RADIUS	YZROT
FEFT	RIGHT	ZLARGE
LENGTH	SCALE	ZSMALL
MINUS	SMALL	ZXPLAN
NEGX	TANTO	ZXROT
NEGY	TIMES	3PT2SL
NEGZ	TRANSL	4PT1SL
NOX	UNIT	5PT
NOY	XLARGE	INTERC
NOZ	XSMALL	SLOPE
IN	MIST	RED
OUT	TAPKUL	GREEN
ALL	STEP	BLUE
LAST	MAIN	INTENS
NOMORE	SIDE	LITE
SAME	LINCIR	MED
MODIFY	MAXIPM	DARK
MIRROR	REV	CHUCK
START	TYPE	COLLET
ENDARC	NIXIE	AAXIS
CCLW	LIGHT	BAXIS
CLW	FOURPT	CAXIS
MEDIUM	TWOPT	TPI
HIGH	PTSLOP	OPTION
LOW	PTNORM	RANGE
CONST	SPLINE	PSTAN
DECR	RTHETA	CSTAN
INCR	THETAR	FRONT
ROTREF	XYZ	REAR
TO	TRFORM	SADTUR
PAST	NORMAL	MILL
ON	UP	THRU
OFF	DOWN	DEEP
IPM	LOCK	TRAV
IPR	SFM	NORMPS
CIRCUL	XCOORD	CONCRD
LINEAR	YCOORD	GECENT
PARAB	ZCOORD	SC4020
RPM	MULTRD	MILWAK
MAXRPM	XYVIEW	BENDIX
TURN	YZVIEW	DYNPAT
FACE	ZXVIEW	TRW
BORE	SOLID	ECS
BOTH	DASH	CINCY
XAXIS	DOTTED	TRUTRA
YAXIS	CL	PRATTW
ZAXIS	DITTO	FOSDIK
TOOL	PEN	BURG
AUTO	SCRIBE	PROBOG
FLOOD	BLACK	DVLIEG
		SUNTRN

USA Participation in an International Standard Glossary on Information Processing

J. F. TRAUB*
Bell Telephone Laboratories, Inc., Murray Hill, N. J.

1. Background

A considerable number of glossaries in the area of information processing have been produced in the USA in the last ten years [1, 2]. In some cases the glossaries were reworked versions of earlier glossaries, while in other cases major new contributions were made. All told, the glossary effort has cost thousands of man-hours of work.

Several years ago the ASA X3 sectional committee sponsored by BEMA was established to prepare standards for the USA in the information processing field. (See Appendix 1 for the meaning of abbreviations and acronyms. See also [3].) ASA X3.5 was assigned the double scope of advising the other X3.n subcommittees on the establishment of definitions required for their proposed standards and of establishing a standard glossary, pASGIP, for general use.

At the same time there was important British standardization activity. After reworking a number of earlier drafts, the BSI released the "Glossary of Terms Used in Automatic Data Processing," British Standard 3527: 1962. The British effort differed in at least one very important respect from the USA glossaries. It was organized along subject rather than alphabetical lines. This was to have important consequences, as we shall see.

A parallel action to the national standardization activities was the formation of ISO/TC97, which held its first meeting in Geneva in May 1961. ISO/TC97/SC1 (then known as ISO/TC97/WGA) was given the task of providing a multilingual glossary. The secretariat was assigned to the Netherlands. Dr. Ir. M. R. Mantz serves as chairman.

In SC1 each participating nation has one vote. The job of the USA representative is to serve as a liaison between SC1 and USA activity between meetings of SC1. For the meetings of SC1, which take place once every one to two years, a delegation is chosen to represent the USA. The leader of the delegation is often the man who serves as USA representative, but this need not be the case.

At its first meeting, SC1 accepted an offer by the IFIP/ICC Joint Terminology Committee to provide it with a first draft of a multilingual glossary. The chairman of the JTC is G. C. Tootill of the United Kingdom; its members are the representatives of the professional societies of various countries. The USA professional society represented is AFIPS. The subject classified "Glossary of

* USA Representative to ISO/TC97/SC1; Chairman, ASA X3.5.2