

The Automation of Kenner's Spirograph:
A Graphic Application of Mathematics

An Honors Thesis/Creative Project (ID 499)

by

Todd A. Alexander

Thesis Director



Ball State University

Muncie, Indiana

May, 1986

Spring Quarter, 1986

SpColl
Thesis
2489
.Z4
1986
.A44

Mathematics, the non-empirical science par excellence...the science of sciences, delivering the key to those laws of nature and the universe which are concealed by appearances.

Hannah Arendt

If all the arts aspire to the condition of music, all the sciences aspire to the condition of mathematics.

George Santayana

Mathematics possesses not only truth, but some supreme beauty--a beauty cold and austere, like that of sculpture.

George W. Russell

CONTENTS

(I) TECHNICAL PAPER

Provides overview of the Spirograph(TM) design toy and discusses the mathematics used to automate the equipment

(II) USER'S MANUAL

User's guide on how to operate the Cyclograph System

(III) SOURCE LISTING

Complete listing of source code modules in the Cyclograph System

(IV) APPENDICES

- A) Current Spirograph(TM) Instruction Booklet
- B) Original Spirograph(TM) Instruction Booklet

Technical Paper

Table of Contents

Forward	1
Spirograph(TM) Overview	4
The Cyclograph and Software Development . .	6
The System	6
How the Designs are Drawn	6
How the Procedural Language Works .	25

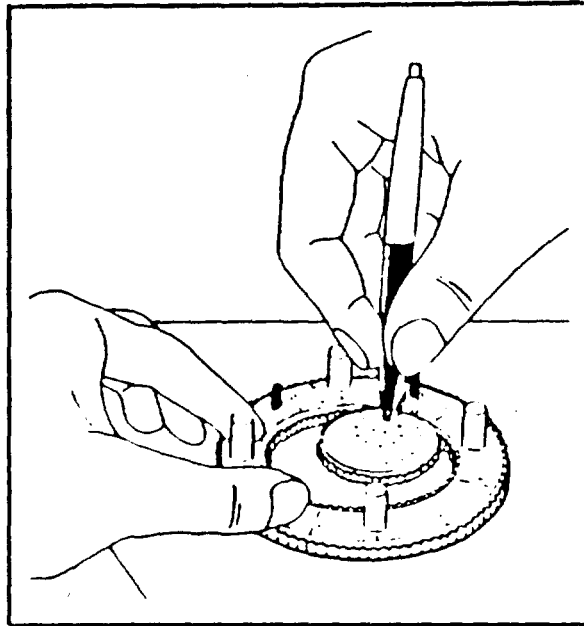
HONORS THESIS/CREATIVE PROJECT

In order to complete the requirements to graduate from the Honors College, a thesis or creative project must be submitted prior to graduation. Having a strong interest in both of my major areas of study, mathematics and computer science, I decided to commit myself to a project dealing in both. Motivated by the word "creative," I began brainstorming for a computer application that would be exciting, challenging, and unique.

I knew very little in the field of computer graphics; however, I knew that I wanted to learn and use this in my project. With my math background, I thought of many things that I could draw that would be exciting. I then remembered an old drawing toy with which I would play for hours. The toy was called a Spirograph and was the brainchild of the engineers at Kenner Products. The toy won numerous prizes in the early 1960's.

The Spirograph(TM) consisted of many small geared wheels and geared rings. By placing a wheel inside one of rings, one could draw a many pointed, curved design. (I would draw for hours. It often required a great deal of time since keeping the pieces in contact while drawing was a feat -- one at which I often failed.) A picture on the back of the box top showed the basic design that each wheel made in each ring. A number on the wheel

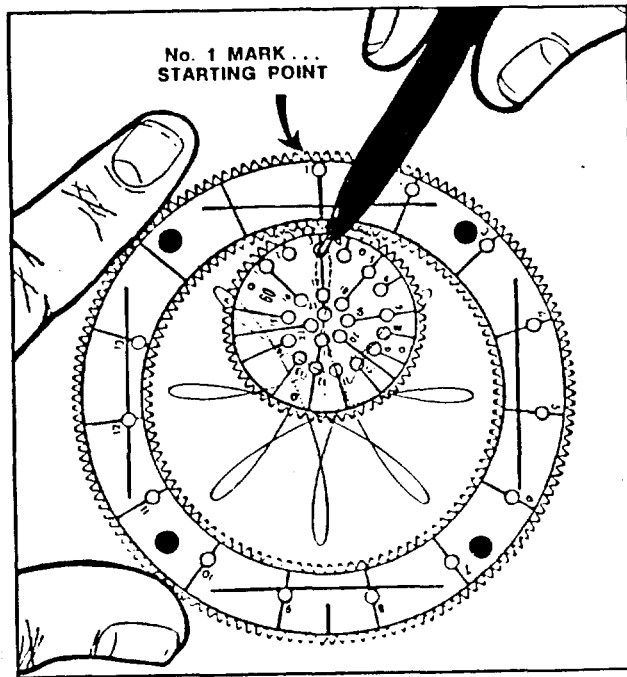
corresponded to the number of teeth it had. Another number was given that told how many points a design had. Some had as many as 105 points, while some had only 2.



The toy is still marketed today, so I purchased a new one. This would be my project. I would make a computer draw these designs on a screen! The idea swelled and I decided I would also develop a language with which one could control the graphing of designs similar to the language used in the instruction booklet. The final project includes (1) this technical paper explaining the how's of the project (thesis), (2) the user's manual that explains how to use the project, and (3) the source code of the system.

It would be helpful if the reader were familiar with

the Spirograph(TM) or even better, had one present. It is my hope that the explanations and examples included are clear and enlightening, even to those not well versed in mathematics or computer science.



SPIROGRAPH(TM) OVERVIEW

SPIROGRAPH(TM) CONTENTS DESCRIPTION

The original Spirograph(TM) consisted of 13 plastic toothed wheels, 2 rings, 2 racks, 4 ball point pens (black, red, orange, and blue,) a fitted storage tray, a cardboard mounting board, mounting pens, and a 16-page color instruction booklet. Today's Spirograph-Plus(TM) is slightly different. The wheels and rings remain the same, but it includes only 1 rack, 3 ball point pens (orange, blue, and red,) and the mounting board and pens have been omitted. Instead, a "ring and rack holder" pierces through the underside of the paper, holding the ring or rack as pegs fit into the holes in it. New additions include 3 wheels which are not circular in form. One is called a "quad" which is shaped like a cross or an "X" shape. Another is the "oval" which is shaped like the outline of a football. And finally, the "triad" is shaped like a triangle constructed of curves, creating a convex form. The "Plus" portion of the new version is from the addition of the "gyro arm" which attaches to the pen and simultaneously draws a skewed design of the original. Two felt tip pens (red and blue) are included for use with the gyro arm.

HOW THE SPIROGRAPH(TM) IS USED TO DRAW DESIGNS

The holder is placed on the backside of the paper. The paper is then pushed onto the holder, allowing the

pegs to pierce the paper. Placing the paper on a flat, smooth surface, a ring or rack is chosen and placed on the pegs of the holder. A wheel is then selected and placed inside or outside of the ring, or on the rack. The pen is placed in the desired hole and the wheel is kept in contact with the ring/rack, moving along the perimeter of the ring/rack for the desired number of revolutions, or until the curve meets itself. This tracing is of the cycloid form. Those curves created with a wheel within a ring are hypocycloids; those with the wheel revolved around the outside of the ring are called epicycloids. All of the cycloids generated with the supplied equipment are prolate epi-/hypocycloids, meaning that the point traced lies within the circle of the wheel. Common cycloids and curate cycloids (tracing a point lying outside of the circle) are not possible since the pen must be constrained within the wheel. Throughout this text, "wheel" will refer to the smaller circle while "ring" will refer to the larger circle in which the "wheel" revolves.

THE CYCLOGRAPH

The Cyclograph System is a computer automated version of the Spirograph(TM). I have given it this name to distinguish it from the trademark Spirograph(TM). It simulates the usage of the contents of the former version, omitting the rack; however, the cyclograph is much more flexible, allowing any size of wheel and ring, not only fixed sizes.

THE CYCLOGRAPH AND SOFTWARE DEVELOPMENT

THE SYSTEM

The Cyclograph System source code was written mainly in VAX Pascal V3.2-57, with two external modules in VAX COBOL V.3.2-42. The object modules were linked under VAX Linker and were designed to run on a VAX 11/780 computer system running under the VAX/VMS 4.2 operating system that supports ReGIS graphics. A VT240 terminal is used to display the graphic designs.

The system allows the user to create designs in an interactive mode, or he or she may use the programming language that supplies a number of advanced functions and allows the storage of design procedures.

DETAILED EXPLANATION:

SOURCE CODE - HOW THE DESIGNS ARE DRAWN

What follows is a detailed description of the mathematical constructs used to develop the software as well as the source code used to implement these ideas. These concepts are usually explained and described more fully with the aid of diagrams and figures. Once an idea has been fully explained, the corresponding source text is presented. An entire source listing is included in the documentation.

In the Beginning

We first begin by mathematically describing, as generally as possible, the curve traced by a point on the

circumference of a wheel as it rotates while revolving within a ring. Let O be the center of the ring (see Figure 1.)

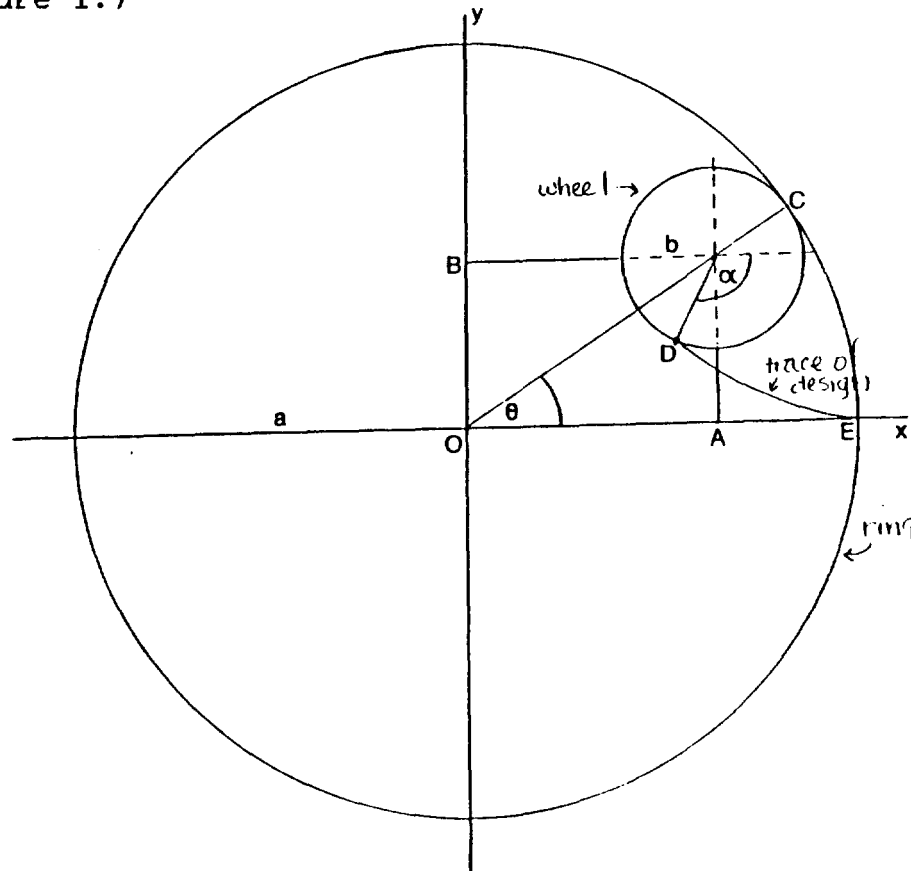


Figure 1. In the Beginning...

The ring has radius a . Let b represent the radius of the wheel. Let C represent the point at which the line from O to the center of the wheel intersects the ring. D is the point on the wheel where our pen is placed. Let E represent the point at which the ring intersects the x axis. Theta is the angle of measurement from the ray OE to the ray OC . Imagine a similar x axis parallel to the first, passing through the center of the wheel. The angle alpha is the measurement from this axis to the ray from the center of the wheel to the point D on the

circumference of the wheel. Observe that

$$\begin{aligned} (1) \quad OA &= (a-b) \cos \theta \\ (2) \quad OB &= (a-b) \sin \theta \end{aligned}$$

Given values for a and b , we can find the center of the wheel. The distance in the x direction is OA ; for the y direction, it is OB .

Since the wheel rolls on the inside of the ring without slipping, we know that

$$(3) \quad CD = CE$$

Also recalling trigonometry, note that

$$\begin{aligned} (4) \quad CD &= b(\alpha + \theta) \\ (5) \quad CE &= a\theta \end{aligned}$$

Recalling (3), α is dependent on the values of θ , a , and b .

$$\begin{aligned} (6) \quad a\theta &= b(\alpha + \theta) \\ (7) \quad a\theta &= b\alpha + b\theta \\ (8) \quad b\alpha &= \theta(a - b) \\ (9) \quad \alpha &= \theta \frac{(a - b)}{b} \end{aligned}$$

We now have determined (given the values for the radius of the wheel, radius of the ring, and angle θ ,) where the center of the wheel lies as well as how much it has revolved.

Determining the Pen Position by Hole Number

After locating the center of the wheel, it is necessary to find the position of the pen given angle α and the distance inward to the center. Allow angle β to be

$$(10) \beta = \theta - \pi/2$$

Imagine the pen being placed on the circumference of the wheel, at point D. (See Figure 2.)

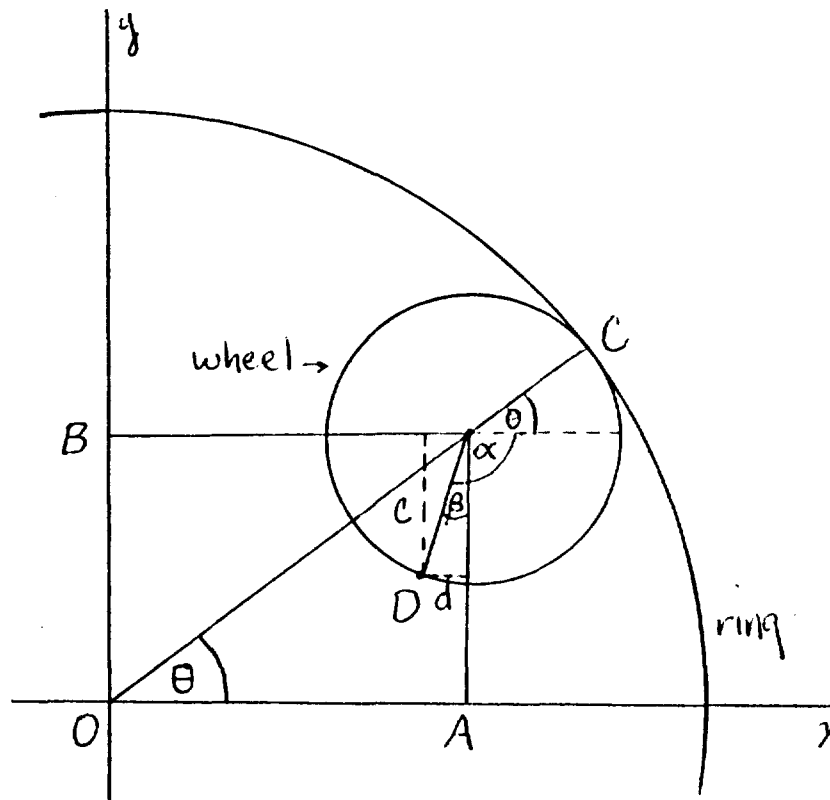


Figure 2. Determining Pen Position by Hole Number

To find this new point from the center of the wheel, we subtract the length c from our original y distance, and length d from the x distance. Hence,

$$(10) \sin \beta = d / b$$

$$(11) d = b \sin \beta$$

$$(12) \cos \beta = c / b$$

$$(13) c = b \cos \beta$$

$$(14) x = OA - d = (a - b) \cos \theta - b \sin \beta$$

$$(15) y = OB - c = (a - b) \sin \theta - b \cos \beta$$

Recall (10). Using trigonometric identities,

$$(16) \sin \beta = \sin(\theta - \pi/2) = -\cos \theta$$

$$(17) \cos \beta = \cos(\theta - \pi/2) = \sin \theta$$

Therefore, (14) and (15) become

$$(18) x = (a - b) \cos \theta + b \cos \theta$$

$$(19) y = (a - b) \sin \theta - b \sin \theta$$

Accounting for the Hole Number

A number of holes exist in a wheel depending on its size. This exact relationship can be defined as follows: the number of holes that a wheel has is seven less than half of the wheel number. Or more mathematically stated:

$$(20) \text{Max Hole Number} = (\text{Wheel Number} / 2) - 7$$

Important Note: The designers of the SPIROGRAPH(TM) used the number of teeth each wheel had as the wheel number. The same holds true for the ring number. I have taken the wheel number (the wheel's "circumference" measured in "teeth") and used it as the radius b . In other words, when the user specifies wheel 24, I use 24 for b , the wheel's radius. Hence (20) becomes

$$(21) \text{Max Hole Number} = (b / 2) - 7$$

These holes are evenly distributed from just within the outside edge of the wheel to just short of the wheel's center. (A hole in the center of the wheel would create an uninteresting circle as the wheel revolved within the ring.) The higher the hole number, the closer the hole is to the center of the wheel. Since the holes do not span the entire radius b of the wheel, the distance was shortened by 5%, or

$$(22) \text{Radius} = (.95)b$$

This new radius is then divided by the number of holes

that appears on the wheel. (Recall (21).) Multiplying this value by the hole number approximates the distance from the edge of the wheel toward the center. Let e represent this determined distance: (see Figure 3)

$$(23) e = (\text{Radius} / \text{Numhole}) * h$$

where h represents the number of the hole chosen, and Numhole represents the number of holes on the specific wheel, see (21).

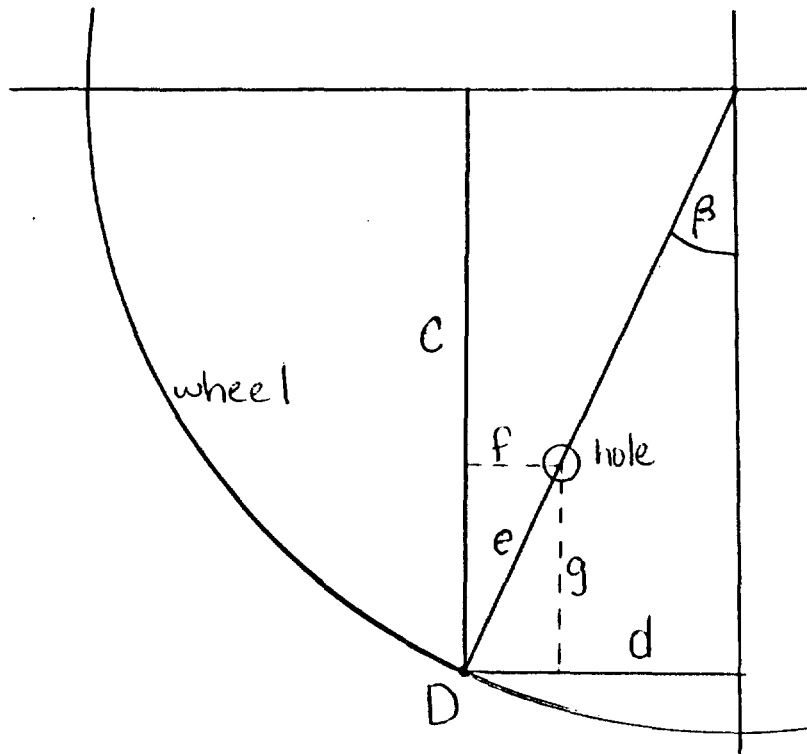


Figure 3. Accounting for the Hole Number

For instance, wheel 24 has $24/2 - 7$, 5 holes. Its adjusted radius is $(.95)(24)$ or 22.8. This value is then divided by 5, yielding 4.56. If hole 5 is chosen, then the distance from the edge of the wheel is the full $e = 22.8$. (Because of using the convention above, the units

of these lengths are in "teeth".) However, if hole 2 is selected, closer to the edge, 2/5 of the distance is used, or $e = 2(4.56) = 9.12$.

Now if we add the x and y components of this distance to our equations (18) and (19), we will have positioned the pen according to the hole number. Let f and g represent the x and y components of e, respectively,

$$(24) f = e \sin \beta$$

$$(25) g = e \cos \beta$$

Adding these to (17) and (18) gives us

$$(26) x = (a - b) \cos \theta + b \cos \theta + e \sin \beta$$

$$(27) y = (a - b) \sin \theta - b \sin \theta + e \cos \beta$$

The Hole is Larger than the Pen

We are not finished. If we truly want to simulate all of the factors contributing to the creation of designs using the Spirograph(TM), we must take into account the fact that the pen moves within its hole. We assume that the contact of the pen in the hole is relative to the contact of the wheel on the ring. That is, the angle made from the center of the hole to the point of contact with the pen is theta. See Figure 4.

Let j represent the distance from the center of the hole to the point of contact of the pen. Measuring the hole's circumference in "teeth", we will apply the convention of using the circumference in teeth as the radius and assign j a value of 3. (This is just an approximation and is surprisingly close when comparing manual drawings to computer generated ones.)

Again, as before, we can break this vector into x and y components. See Figure 4.

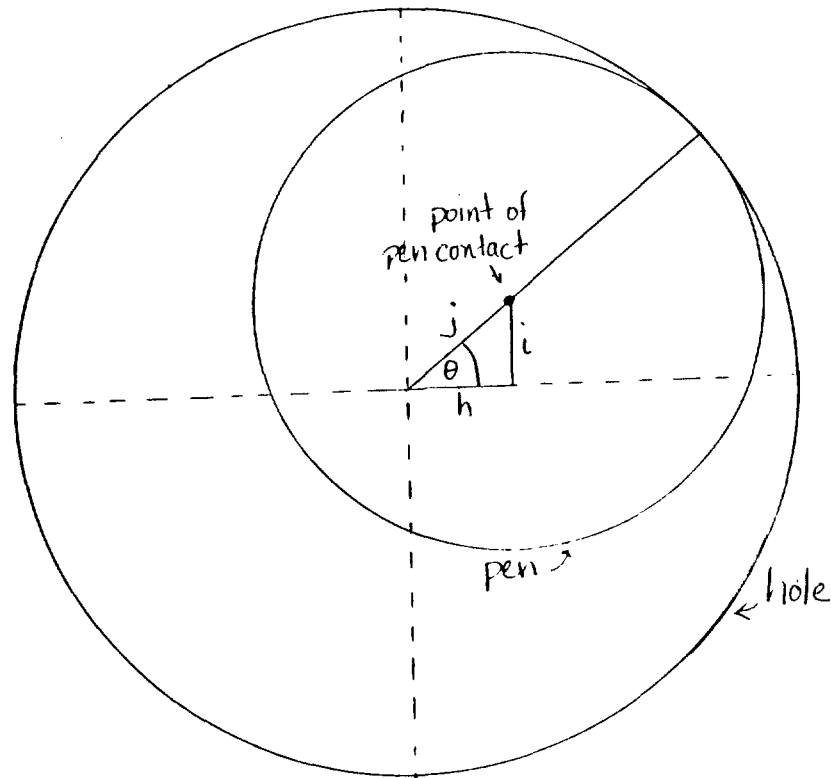


Figure 4. The Pen Moves in the Hole

$$(28) \quad h = j \cos \theta$$

$$(29) \quad i = j \sin \theta$$

Adding these to (17) and (18) gives us

$$(30) \quad x = (a - b) \cos \theta + b \cos \theta + e \sin \beta + j \cos \theta$$

$$(31) \quad y = (a - b) \sin \theta - b \sin \theta + e \cos \beta + j \sin \theta$$

Note the common terms. Combining, we have:

$$(32) \quad x = (j + a - b) \cos \theta + b \cos \theta + e \sin \beta$$

$$(33) \quad y = (j + a - b) \sin \theta - b \sin \theta + e \cos \beta$$

We are now ready to present source code that implements our combination of ideas: 1) pen which revolves in hole with radius j , 2) in rotating wheel with radius b , 3) which revolves θ degrees, 4) in ring with radius a .

```

x := (a-b+j)*Cos(Theta) + b*Cos(Theta*(a-b)/b) + e*Sin(Beta);
y := (a-b+j)*Sin(Theta) - b*Sin(Theta*(a-b)/b) + e*Cos(Beta);

```

These two Pascal statements are the "heart" of the system. They look very much like the expressions (32) and (33), and they should. They are the basic equations from the constructions used to plot designs. (To see the statements in complete context, consult the source code listing in its entirety.)

What about Theta?

The question arises, what values for theta do we use? If we let theta range from 0 to 360 degrees, we have drawn one revolution of the wheel's center. The curve is periodic, but did it meet itself?

The answer lies in the ratio of the radius of the ring to the radius of the wheel. For example, if we used a wheel and ring of radius 48 and 96, respectively, we would expect the curve to meet itself after one revolution. Intuitively, a wheel with half the radius of the ring would travel in the ring, rotating twice. The key lies in the fact that 96 is a multiple of 48, or 96 divided by 48 is a whole number. Hence the wheel, after one revolution around the ring, will line up exactly where it began. If the ratio is not a whole number, say 96 divided by 30, and we let the wheel revolve until it does end up where it began, we have successively added the ratio to itself until a whole number is achieved. Once a

whole number is reached, the wheel is back where it began. The ratio 96 divided by 30 is 3.2. Multiples of the ratio are 3.2, 6.4, 9.6, 12.8, 16. We would meet our design again in 5 revolutions around the ring. The answer for this example would be to let theta begin at 0 and travel $360 * 5$ degrees. To find the number of cycles necessary to complete a design, the following Pascal procedure was used:

```

00304 Procedure Find_closure_cycles;
00305   (Calculates number of cycles necessary to complete a design).
00306   Begin
00307     Ratio := a/b;
00308     Cycles := 0;
00309     M := 1;
00310     While (Cycles = 0) Do
00311       Begin
00312         If abs(trunc(M * Ratio) - (M * Ratio)) < 0.0001 then Cycles := M;
00313           M := M + 1;
00314       End;
00315   End;

```

The variable Ratio in line 307 was assigned the value of the ratios of the ring to the wheel. M in line 309 counts from 1 to whatever is necessary to satisfy the IF condition in line 312. We allow M to count, while watching the product of M with the Ratio. Line 312 tests to see if the product is a whole number (or at least necessarily close to a whole number.) Once the whole number is reached, we terminate counting and assign the variable Cycles the value of M which when multiplied times the ratio, yields a whole number. Then, in general, we

allow theta to start at 0 and end at $360 * \text{Cycles}$.

Skipping Teeth

The way we have our equations now, the wheel begins only on one place on the ring -- where the angle is 0, or the intersection of x axis and the ring. (See Figure 1.) In reality, we could start the wheel anywhere within the ring. We must provide a means of starting the wheel elsewhere. We do this by rotating the original design in fixed increments. We cannot place the wheel in the ring without meshing of teeth, hence these increments are angles corresponding to the teeth on the ring. (Often design instructions explicitly say "move wheel one tooth to the right" which is exactly the same as rotating the design.) To find what angle is needed to represent the skipping of a tooth, we divide 360 degrees by the number of teeth the ring has, or more precisely, the ring number a . (How incredibly convenient.) Multiplying this increment by the number of teeth the user wishes to skip yields the angle through which to rotate the design. Consider the following Pascal procedure which does precisely this:

```

00438 Procedure Move_teeth;
00439   {Rotate design in increments of "Teeth"}
00440   Begin
00441     Find_number;
00442     Rotation := Rotation + Num * (2*3.1415/a);
00443   End;
```

The variable Rotation holds the values of the angle (in

radians) used to rotate the design. (Note that all degree values have to be converted to radians since the built-in functions of Pascal does not use degree units. See line 384 of the source text.) The variable appears on the right side of the assignment statement as well; such an angle is being accumulated so that if the system is told to skip 5 teeth, 5 teeth are skipped from the current rotation of the wheel.

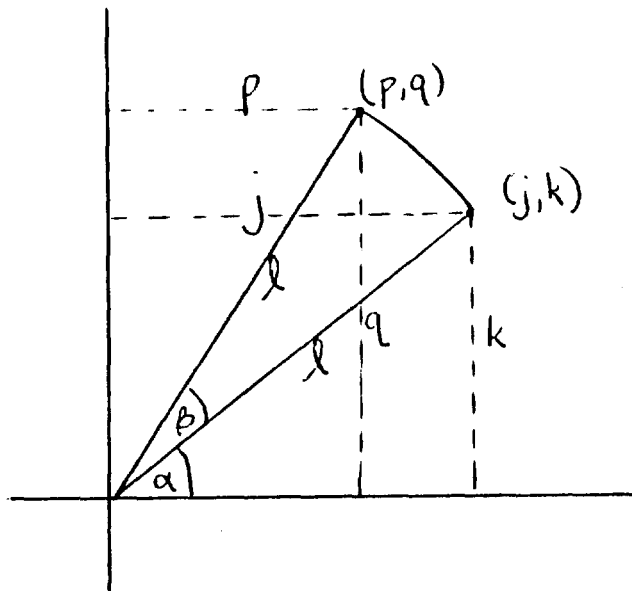


Figure 5. Rotating β Degrees

Consider Figure 5. We want to rotate point (j, k) around the origin through angle beta. We know the values for j and k , and using the distance formula, we can find l .

$$(34) \quad j = l \cos \alpha$$

$$(35) k = l \sin \alpha$$

$$(36) l = \sqrt{j^2 + k^2}$$

Similarly, p and q are as follows:

$$(37) p = l \cos (\alpha + \beta)$$

$$(38) q = l \sin (\alpha + \beta)$$

We know the values of β and l , but we do not know the value of alpha.

$$(39) \alpha = \cos (j / l)$$

This is simple enough for a hand-held calculator, but not for Pascal. Pascal has no built-in inverse trigonometric functions, and instead of formulating one, we can use the identity

$$(40) \cos (\alpha + \beta) = \cos(\alpha) \cos(\beta) - \sin(\alpha) \sin(\beta)$$

since we know the value of $\cos(\alpha)$. In a completely similar manner, the same holds true for $\sin (\alpha + \beta)$.

Hence, (37) and (38) become

$$(41) p = l [\cos(\alpha) \cos(\beta) - \sin(\alpha) \sin(\beta)]$$

$$(42) q = l [\sin(\alpha) \cos(\beta) + \cos(\alpha) \sin(\beta)]$$

Examine the source code below:

```

00388   Length := Sqr(x*x + y*y);
00389   cos1 := (x/length);
00390   sin1 := (y/length);
00391   Sinr := sin(rotation);
00392   Cosr := cos(rotation);
00393   cossum := Cosr*cos1 - Sinr*sin1;
00394   sinsum := Sinr*cos1 + Cosr*sin1;

```

The variables x and y correspond to j and k above. Length replaces l and is calculated in line 388. Rotation

replaces the angle beta and is calculated as described above. The series of equations from 388 to 394 are equivalent to (41) and (42) above, except that the multiplication of l (Length) has been deferred.

What Do X and Y Mean Now?

The value of x and y have not been reassigned as yet. Lines 395 and 396 place new values into x and y. These new values are modified by what seems garbage at first:

```
00395  x := - length * sinsum * 2.9 + 500;
00396  y := length * cossum * 2.9 + 300;
```

First of all, we are taking what originally would be assigned to x and assigning it to y, and vice-versa. Notice how the variable Cossum would be used in the x-direction but is now in y's statement. Notice also the negative sign that appears in x's statement. What we have done is to place the beginning of the drawing at the top of the screen by rotating the design 90 degrees. Plotting (-y,x) instead of (x,y) will accomplish this, and this is what we have done. Note Figure 6.

What about the * 2.9 + 500, etc.?

Our screen has 1000 graphic positions in the x direction and 600 in the y direction. (These numbers are not equal since the screen is not square.) The positions are labeled as in Figure 7.

If we were to plot our design using our current values of

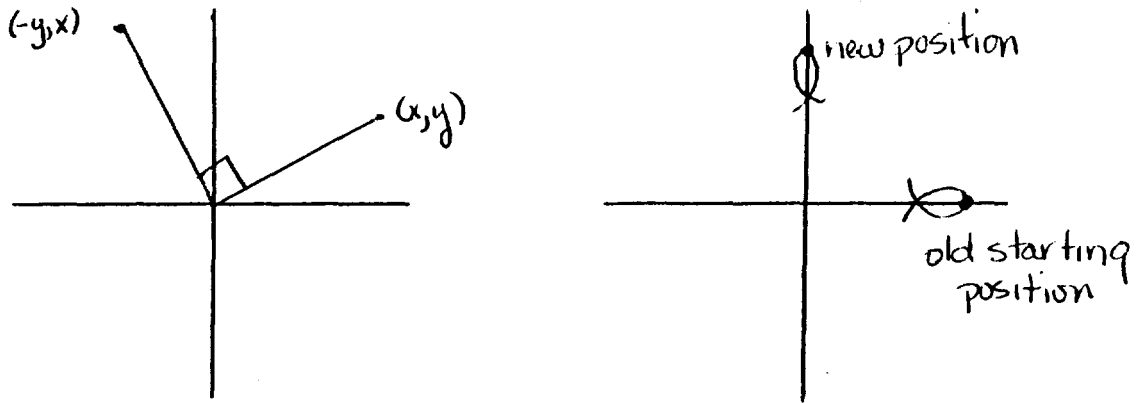


Figure 6. Aligning Starting Position At Top

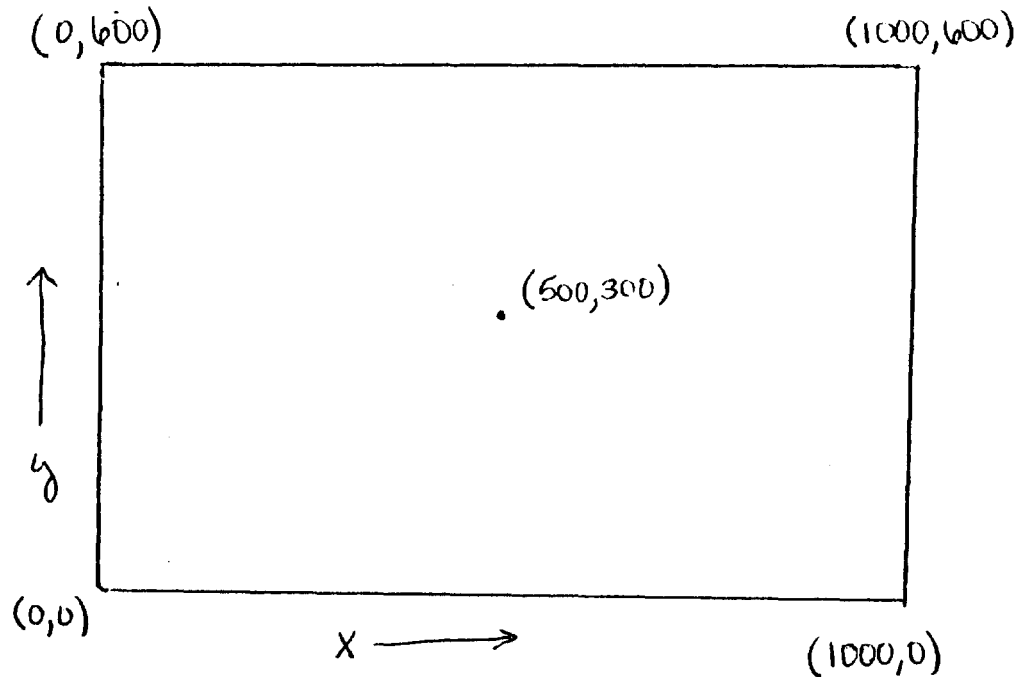


Figure 7. Graphic Screen Locations

x and y, only one-quarter of the design would appear in the lower left corner of the screen. (See Figure 1. Imagine placing point 0 at (0,0) in Figure 7.) It would be

small, with a normal maximum radius of 105. Hence, we shift the origin of our design to the middle of the screen by adding 500 to the x value, and 300 to the y value. We magnify our design by a factor of 2.9 to just fit in the screen with a radius of 105. Now our design is nicely centered with drawing beginning at the top of the screen.

Interfacing with ReGIS

In order to use the graphing capabilities of a terminal, the system must somehow access commands used to draw points and lines on the display. The Cyclograph System uses the ReGIS RAL package to execute graphic commands. Commands are supplied that clear the screen, change the plotting color, and plot points.

We need only two commands, 1) Move, and 2) Line, that will allow us to construct the design on the screen. The Move command does not draw on the screen. It places the graphic cursor at the location specified by the values given it. The Line command connects the current graphic cursor location to the location specified with a line. We could plot a series of points close enough to make the series appear as a line, but this would take quite a while for the computer to draw.

Take a close look at lines below.

```
00383   Theta := Theta2 * 5;
```

```
00397   If Theta2 <> 0 then Line(x,y);  
00398   Move(x,y)
```

Line 383 makes theta skip in increments of 5 degrees. This quickens the drawing time necessary to complete the design.

Line 397 basically means: if this is not the first point, connect the point (x,y) to the last point plotted. Line 398 makes this point, (x,y), the last point plotted for the next time around.

The Hypocycloid

As theta sweeps from its starting and ending limits in steps of 5 degrees, the design is drawn on the screen. We should now have a good understanding of the procedure named Gograph, which appears fully as:

```

00361      1 0 Procedure Go_graph;
00362 C 1 0 {Graphs wheel within a ring, according to cycles, a, and b}
00363      1 1 Begin
00364      1 1   Radius := 0.95 + b;
00365      1 1   numhole := Trunc(b/2 - 7 + 0.5);
00366      1 1   If numhole < 5 then numhole := 5;
00367      1 1   e := (Radius/Numhole) * h;
00368      1 1   j := 3;
00369      1 1   If Immediate_mode then Find_number;
00370      1 2   If Num = 0 then Begin
00371      1 2       Find_closure_cycles;
00372      1 2       Cycles := Cycles * 2;
00373      1 2       Strt := 0;
00374      1 2       Stp := Cycles;
00375      1 2       End
00376      1 2   else Begin
00377      1 2       Find_loop_cycles;
00378      1 2       Cycles := Cycles * 2;
00379      1 2       Stp := Cycles + Strt;
00380      1 1   End;
00381      1 1   For Theta2 := Trunc(36 * Strt) to Trunc(36 * Stp)
00382      1 1   Do Begin
00383      1 2       Theta := Theta2 * 5;
00384      1 2       Theta := Theta + (3.1415/180);
00385      1 2       Beta := Theta*(a-b)/b - 3.1415/2;
00386      1 2       x := (a-b+j)*Cos(Theta) + b*Cos(Theta*(a-b)/b) + e*Sin(Beta);
00387      1 2       y := (a-b+j)*Sin(Theta) - b*Sin(Theta*(a-b)/b) + e*Cos(Beta);
00388      1 2       Length := Sqr(x*x + y*y);
00389      1 2       cos1 := (x/length);
00390      1 2       sin1 := (y/length);
00391      1 2       Sinr := sin(rotation);
00392      1 2       Cosr := cos(rotation);
00393      1 2       cossum := Cosr*cos1 - Sinr*sin1;
00394      1 2       sinsum := Sinr*cos1 + Cosr*sin1;
00395      1 2       x := - length * sinsum * 2.9 + 500;
00396      1 2       y := length * cossum * 2.9 + 300;
00397      1 2       If Theta2 <> 0 then Line(x,y);
00398      1 2       Move(x,y);
00399      1 1   End;
00400      1 1   Strt := Stp;
00401      0 0 End;

```

Conversions to radians and the restrictions placed on theta are determined by the user's commands and are

therefore not explained in full detail. This includes lines 369 through 380 and line 400.

The necessary information is entered by the user using one letter commands followed by any necessary value. This consists of acquiring information from the user via the keyboard and assigning the appropriate values to variables. It is hardly interesting and is therefore not discussed.

DETAILED EXPLANATION:

SOURCE CODE - HOW THE PROCEDURE LANGUAGE WORKS

The Procedural Language

The procedure language commands allow the user to create complex designs and store the instructions needed to do so. The command set is quite full, allowing most functions that are described in the instruction booklet supplied with the Spirograph(TM). Advanced functions are included which create designs which are not possible using the Spirograph(TM) equipment. See the User's Manual for a complete listing of commands and how they are used.

The Procedure File Format

All procedural commands are created and stored externally in a file specified by the name PROCFILE.DAT. The editor used to create and edit this file is EDT. The commands must be in capital letters and only one command is allowed per line in the file. Blank lines are permitted. The line may be placed anywhere and contain as many embedded blanks as desired.

Example Procedure File

```
1)  START EXAMPLE1
2)   RING 105
3)   WHEEL 24
4)   HOLE 85
5)   DO 5 TIMES
6)   GRAPH
7)   + 1 HOLE
8)   MOVE 2 LEFT
9)   END
10)  RUN EXAMPLE2
```

```

11) STOP
12)
13) START EXAMPLE2
14) WHEEL 30
15) GRAPH
16) STOP

```

The above is an example of a typical procedure file; however, the line numbers to the right have been added for explanation purposes and do not appear in the actual file. The cyclograph first reads the file and stores the command lines in an array called Commlist, removing any blank lines. The file, now stored in memory, looks like:

```

1) START EXAMPLE1
2) RING 105
3) WHEEL 24
4) HOLE 85
5) DO 5 TIMES
6) GRAPH
7) + 1 HOLE
8) MOVE 2 LEFT
9) END
10) RUN EXAMPLE2
11) STOP
12) START EXAMPLE2
13) WHEEL 30
14) GRAPH
15) STOP

```

The procedure that accomplishes this is called Readfile and appears as:

```

00525 Procedure Read_file;
00526 Begin
00527   Reset(Procfile);
00528   k := 1;
00529   While not eof(Procfile) and (k < 201) Do
00530     Begin
00531       Readln(Procfile,CommLine);
00532       If commline <> ' ' then
00533         Begin
00534           Commlist[k] := CommLine;
00535           k := k + 1;
00536         End;
00537     End;
00538 End;

```

The next step in interpreting the file is to separate it into individual words. The procedure that accomplishes this called Buildwordlists and can be found on page 12 of the source listing. The Pascal code basically separates contiguous strings of characters into column lists. The first contiguous string (which we would call a "word" and will do so in the future) is placed in the first column list; the second is placed in the second list, etc. Our procedures now appear as

	1st	2nd	3rd	4th
	-----	-----	-----	-----
1)	START	EXAMPLE1		
2)	RING	105		
3)	WHEEL	24		
4)	HOLE	85		
5)	DO	5	TIMES	
6)	GRAPH			
7)	+	1	HOLE	
8)	MOVE	2	LEFT	
9)	END			
10)	RUN	EXAMPLE2		
11)	STOP			
12)	START	EXAMPLE2		
13)	WHEEL	30		
14)	GRAPH			
15)	STOP			

Syntax Check

We now run down the list of lines, one by one, and check the following items:

- 1) Is the first word a valid command?
- 2) If the command requires a value, is it present? Valid?
- 3) If it requires a third word, is it present? Valid?
- 4) Does anything appear in column 4? (Notice that no commands are longer than 3 words in length.)

If any of the first 3 are false, an error message is displayed, telling the user what he or she has incorrect or missing. If number 4 is true, extraneous information was placed on the command line, or spaces inadvertently appear where they should not.

Some special things also happen during this initial check. If the first word of the line is START, the second word is placed in a list of valid procedure names, regardless of its content. The line number corresponding to this statement is also stored in a list. If the first word of line is RUN, the second word is placed in a list of called procedures, similar to the process used with the START statement. Paired statements such as START/STOP and DO/END are checked to see if each is paired correctly. If not, the user is informed of an error of this type.

After the initial check, we have a list of procedure names, where they are located, and a list of called

procedures. We now go through the list of called procedures and check to see if they are in the list of valid procedure names. If not, we inform the user of this.

An error summary is shown to the user. If there are any errors in the procedure file, he or she is not allowed into the programming mode of the system. The file must be corrected before it may be executed. This eliminates the possibility of "run-time" errors because of disregard for syntax rules.

Once no errors are found in the procedures, the user may "run" his or her procedures in the programming mode. By supplying a valid procedure name, the procedure is interpreted and the design is produced on the terminal.

The Interpreter

This supplied procedure name is located in the list of valid procedure names. If it is not present, the user is told so; however, if it is present, the first line after the START statement is interpreted and executed. Many of the same Pascal routines used in the immediate mode of the system, especially Gograph, are used in the programming mode. And so the interpreter goes, down the list of commands broken into words, doing the necessary tasks as the user dictates them, until special statements are found as described below. These tasks are simple and can be examined in the source listing.

Execution: Flow of Control

Special commands, the END and STOP commands, cause the interpreter to change what line it is currently "executing." If a DO command is encountered, stating that the following block of commands should be repeated, the current line is placed in a special storage list called a stack. The number of times to repeat the block is placed on a different stack. There can be many values stored on a stack. The unique quality that this storage structure has is that the last value placed on it is the only one which can be retrieved. This is needed if a number of DO commands are nested within one another; in this way, only the inner-most DO block will be dealt with at one time. The interpreter travels down the list until an FND statement is encountered. Once encountered, the stack holding the number of repetitions is checked. If more repetitions are needed, the interpreter subtracts one from this number and skips back to the line following the DO statement. Once the number of repetitions reaches 0, the interpreter removes the line number and repetition value from the stack. Perhaps another DO statement has its line number on the top of the stack now, waiting on an END statement.

The RUN statement tells the interpreter to skip to the line number associated with the given procedure name. In a similar manner, the RUN statement places a line number on the stack, telling the interpreter where to return after reaching a STOP statement in the called

procedure.

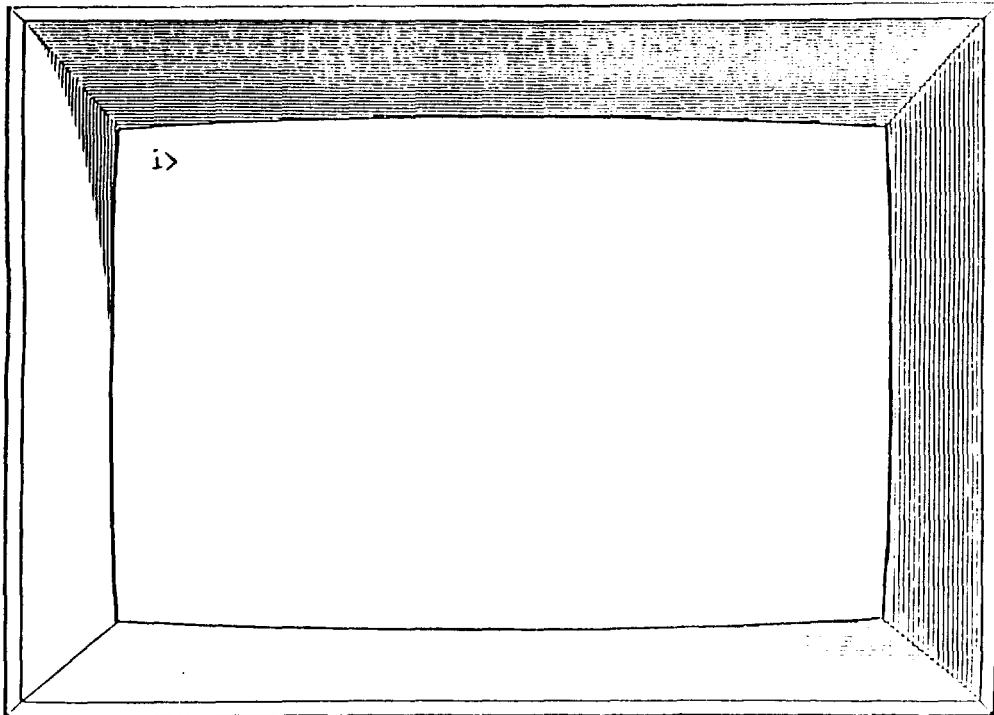
The stacks defined in the system hold only 20 values; therefore, only 20 levels of nesting are possible. Twenty levels should be sufficient, but nesting beyond this limit will cause a fatal error.

"Execution" of the procedure continues until a STOP statement is encountered while the interpreter is not performing a RUN or DO statement. (In other words, when it is not embedded in a level of repetition.)

An Invitation

Now that a good understanding of the how's of the cyclograph has been grasped, it is my hope that the reader will better appreciate and comprehend what is happening upon the execution of the cyclograph system and that most of the mystery has been dispelled. If the system has not been seen, please feel free to try it. A User's Manual is included in the project.

User's Manual



In order to use the Cyclograph System, the user must make sure the following conditions are met:

- 1) The user logged in under the VAX Cluster system,
- 2) the user has a current copy of CYCLOGRAPH.EXE,
- 3) is using a VT 240 terminal,
- 4) and the terminal is set for a VT 100, using the set-up mode.

After these conditions are met, the user should begin execution by entering RUN CYCLOGRAPH. The screen should appear as above.

This is the immediate mode of the cyclograph. Notice the lower case i which reminds you of this. In the immediate mode, one letter commands are used to specify what size of wheel, ring, etc., and to control the drawing of designs. The immediate mode commands are described below:

IMMEDIATE MODE COMMANDS

Specifying Values

COMMAND	FUNCTION
R n	The R command stands for Ring. Use it to specify the ring number. N represents the ring number and must be an integer.
W n	The W command stands for Wheel. Use it to specify the wheel number. N represents the wheel number and must be an integer.
H n	The H command stands for Hole. Use it to specify the hole number. N represents the hole number and must be an integer.
P x	The P command stands for Pen. Use it to specify the pen color. X represents the color and must be RED, BLUE, or GREEN.

Controlling the Drawing

COMMAND	FUNCTION
G	The G command stands for GRAPH. Use it to begin drawing with the current settings of the hole, wheel, and ring. Drawing will continue until the curve is closed.
G n	The G command followed by a number stands for GRAPH n LOOPS. Use it to draw only n loops of the complete design. N must be an integer.
C	The C command stands for Clear Screen. Use it to erase the screen.

Rotation / Resetting Wheel

COMMAND	FUNCTION
M n	The M command stands for Move. N represents the number of teeth in which to move the wheel to the left. N must be an integer.
Z	The Z command stands for Zero. This command sets the wheel back to the top of the ring, clearing all rotation.

Exiting System / Entering Programming Mode

COMMAND	FUNCTION
E	The E command stands for Exit. Use it to exit the Cyclograph System and terminate execution.
*	Use the asterisk (*) command to switch between immediate and programming modes.

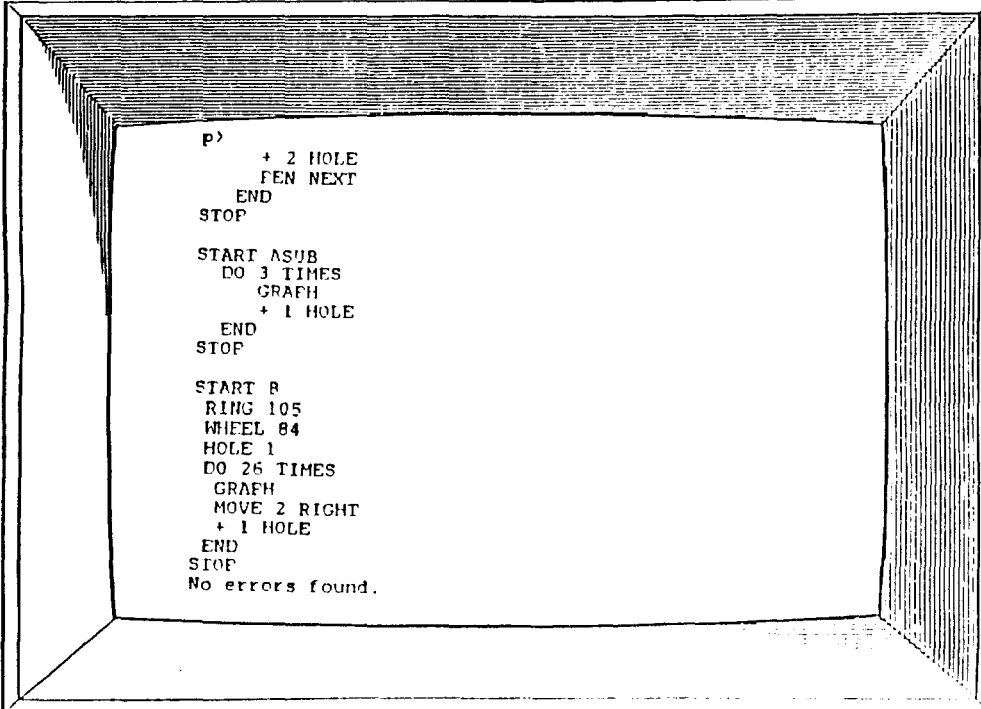
Rotation / Resetting Wheel

COMMAND	FUNCTION
M n	The M command stands for Move. N represents the number of teeth in which to move the wheel to the left. N must be an integer.
Z	The Z command stands for Zero. This command sets the wheel back to the top of the ring, clearing all rotation.

Exiting System / Entering Programming Mode

COMMAND	FUNCTION
E	The E command stands for Exit. Use it to exit the Cyclograph System and terminate execution.
*	Use the asterisk (*) command to switch between immediate and programming modes.

PROGRAMMING MODE COMMANDS



```
P>
  + 2 HOLE
  FEN NEXT
  END
  STOP

  START ASUB
  DO 3 TIMES
  GRAFH
  + 1 HOLE
  END
  STOP

  START R
  RING 105
  WHEEL 04
  HOLE 1
  DO 26 TIMES
  GRAFH
  MOVE 2 RIGHT
  + 1 HOLE
  END
  STOP
No errors found.
```

Once you have entered the programming mode, your procedures are loaded from the procedure file and checked for errors. If errors exist, the immediate mode will be invoked and you will not be permitted to run your procedures. You must edit your files, correcting those things which caused errors. Consult the rules below if you are having difficulty. The error messages are self-explanatory. If all is well, your screen should appear as above with the message "No errors found." Simply enter the name of the procedure that you wish to execute.

You must return to the immediate mode to exit the

system. Remember that the asterisk (*) command does this.

CREATING A PROCEDURE FILE

0) Introduction --

The Syntax Rules

- a) Only one command is allow per line; however, it may be placed anywhere on the line.
- b) Blank lines are ignored, and may be placed anywhere in the source text.
- c) Blank characters are ignored between keywords and values in the source text.
- d) The file in which to store the commands should be called PROCFILE.DAT in your directory.

Guide to Explanations

- a) The symbol | means "or" and is used to separate items in a list of choices.
- 1) START / STOP commands -- Writing a design procedure
- Named procedures begin with the keyword START, followed by the name of the procedure. This name is supplied by the designer and may be any contiguous string of characters. The procedure ends with the keyword STOP. More than one procedure may be stored in procedure file; however, the file must contain at least one procedure.

Example:

START DESIGNA

.

.

. Procedural commands

.

.

STOP

Note: DESIGNA is the name given
to this procedure by the designer.

2) Procedural Commands -- Specifying Values

a) RING n -- Specifying the ring number

Use the RING command to specify the ring number (size). If no number is selected, the default value of 24 will be used. The ring value will remain unchanged until another execution of the RING command. The value n may be any positive integer.

b) WHEEL n -- Specifying a wheel number

Use the WHEEL command to specify a wheel number (size). If no number is selected, the default value of 96 will be used. The wheel value will remain unchanged until another execution of the WHEEL command. The value n may be any positive integer.

c) HOLE n -- Specifying the hole number

Use the HOLE command to specify the hole number. If no number is selected, the default value of 1 will be used. The HOLE value will remain unchanged until another execution of the HOLE command. The value n may be any positive integer up to 7 less than the wheel number divided by 2.

d) PEN x -- Specifying the pen color

Use the PEN command to specify a pen color. If no color is selected, the default value GREEN will be used. The pen color selected will remain in effect until the execution of another PEN command. The value of x may be BLUE | GREEN | RED | NEXT. The specification NEXT will set the color to the next color in the alphabetized list of colors. The alphabetized list is BLUE, GREEN, and RED. In other words,

if GREEN is the current color, COLOR NEXT would set the color to RED. (RED wraps back to BLUE in the list.)

3) Procedural Commands -- Graphing the Current Settings

a) RESET -- Resetting wheel or gear to starting position

Use the RESET command to align the wheel or gear with mark 1 on the ring. Not doing so before drawing will allow drawing to begin where the last design left off.

b) GRAPH -- Drawing a closed design

Use the GRAPH command to draw a design using the current values. Drawing will continue until the design meets itself.

c) GRAPH n LOOPS -- Drawing a partial design

Use the GRAPH n LOOPS command to draw a design using the current values. Drawing will continue for only the specified number of loops, determined by the value n. N may be any positive integer.

4) Procedural Commands -- Special modifications while drawing

a) MOVE n LEFT / MOVE n RIGHT -- Shifting teeth

Use the MOVE n LEFT or MOVE n RIGHT command to move the wheel or gear the specified number of teeth to the left or right. N represents the number of teeth and may be any positive integer.

b) o n WHEEL / o n HOLE / o n RING

-- Modifying Pseudovariables

Use the o n WHEEL | HOLE | RING command to modify the values of the wheel, hole, or ring respectively. The letter o represents an operator and is from the set { + , - , * , / } where + represents addition, - represents subtraction, * represents multiplication, and / represents division, while n may be any real number.

Example:

For example, if the designer wished to add 1 to hole value and divide both the wheel and ring values by 2, he or she would write:

```
+ 1 HOLE
/ 2 RING
/ 2 WHEEL
```

Note: Only the values of the ring, wheel, or hole may be changed in this manner.

5) Procedural commands -- Controlling execution flow

a) DO n TIMES / END -- Repetition structures

Use the DO n TIMES to signify the beginning of a repeating block of commands, where n is a positive integer specifying the number of times to repeat the block. END signifies the termination of the block. Repetitive blocks may be nested. See the example below:

Example:

```
DO 5 TIMES
.
DO 3 TIMES
.
.
END
END
```

b) RUN x -- Calling defined procedures

Use the RUN x to call another procedure within a procedure. X represents the procedure name and must be defined within the current procedure file. Procedures may not call themselves. To see an example, consult the example procedures that follow.

Procedure Examples

Source Commands

```

START A
  WHEEL 30
  RING 105
  HOLE 1
  DO 4 TIMES
    RUN ASUB
    + 15 WHEEL
    + 2 HOLE
  PEN NEXT
  END
STOP

```

```

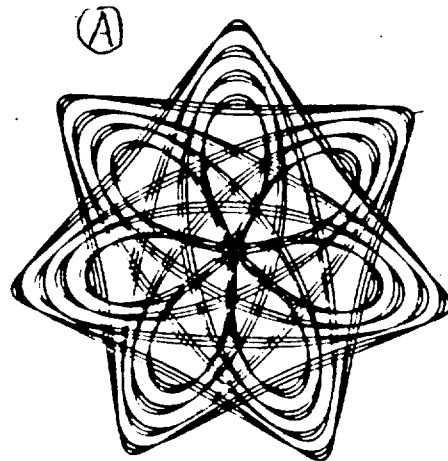
START ASUB
  DO 3 TIMES
    GRAPH
    + 1 HOLE
  END
STOP

```

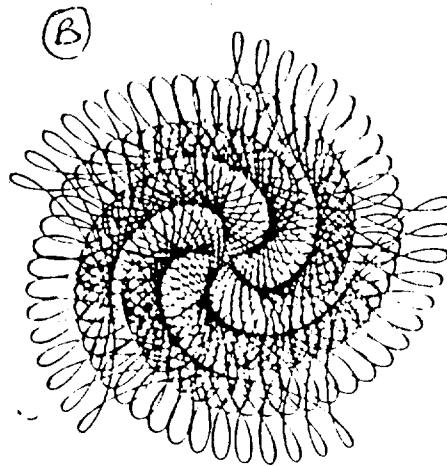
```

START B
  RING 105
  WHEEL 84
  HOLE 1
  DO 26 TIMES
    GRAPH
    MOVE 2 RIGHT
    + 1 HOLE
  END
STOP

```

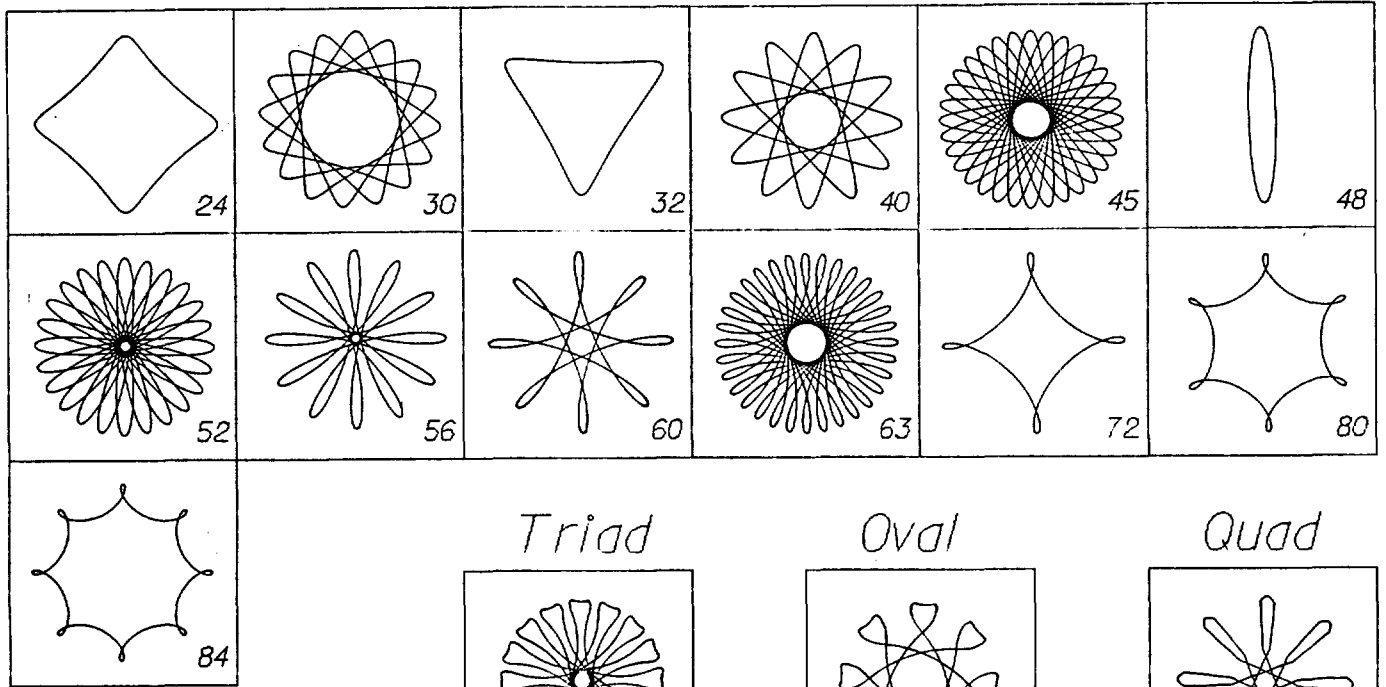


$\frac{150}{105}$ • (30) • 1-2-3 (45) • 6-7-8
 (60) • 11-12-13 (75) • 16-17-18



$\frac{150}{105}$ • (84) • 1 thru 26 moving two teeth
 right every hole

Wheels

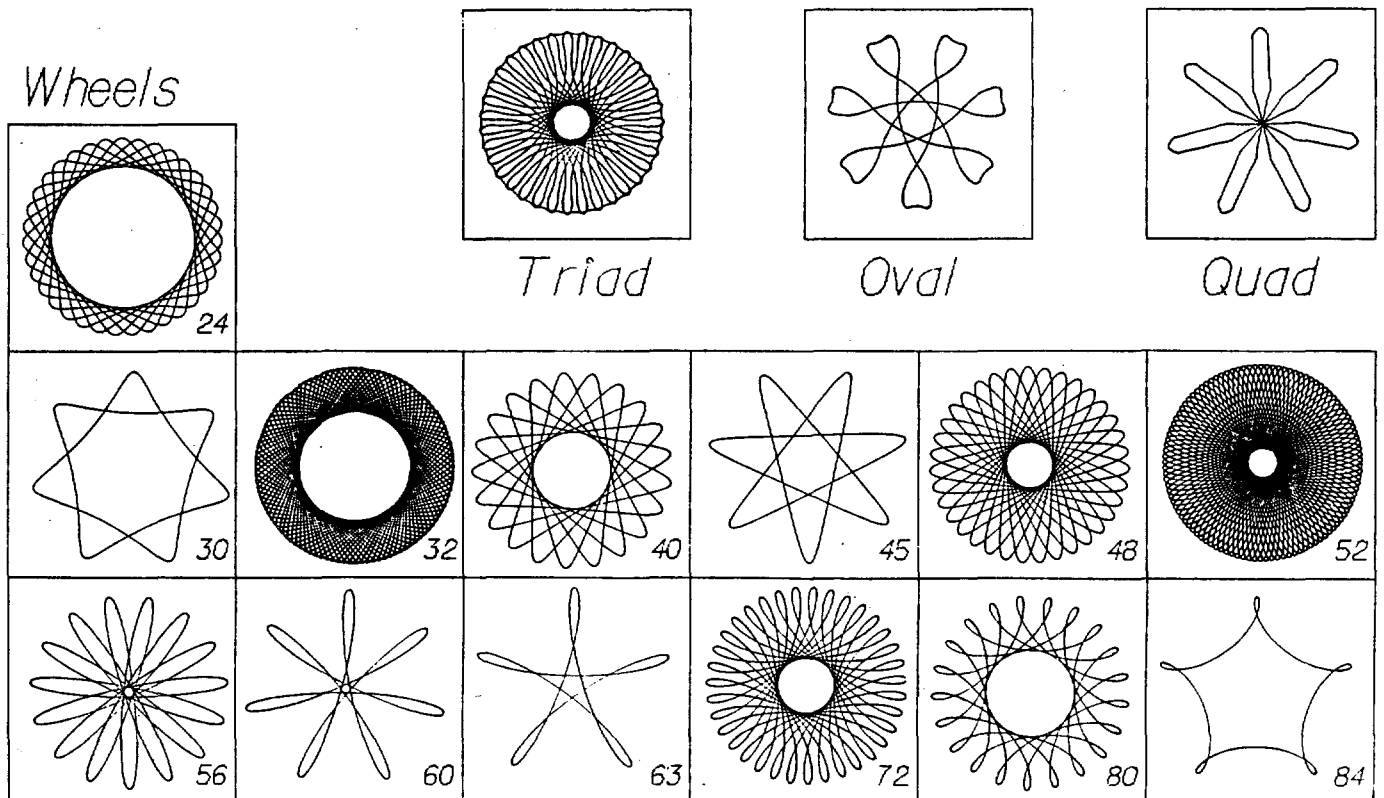


The number beside the wheel design indicates the wheel number.

The above designs were made with ring 144/96.

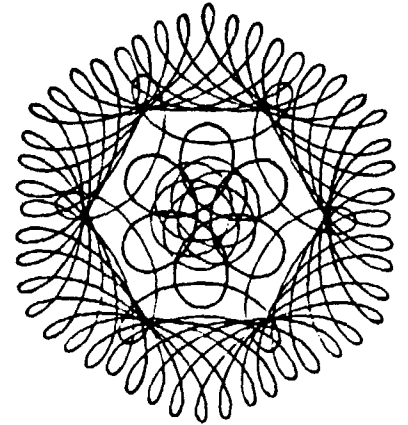
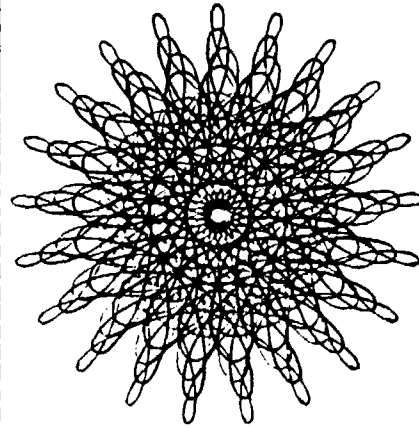
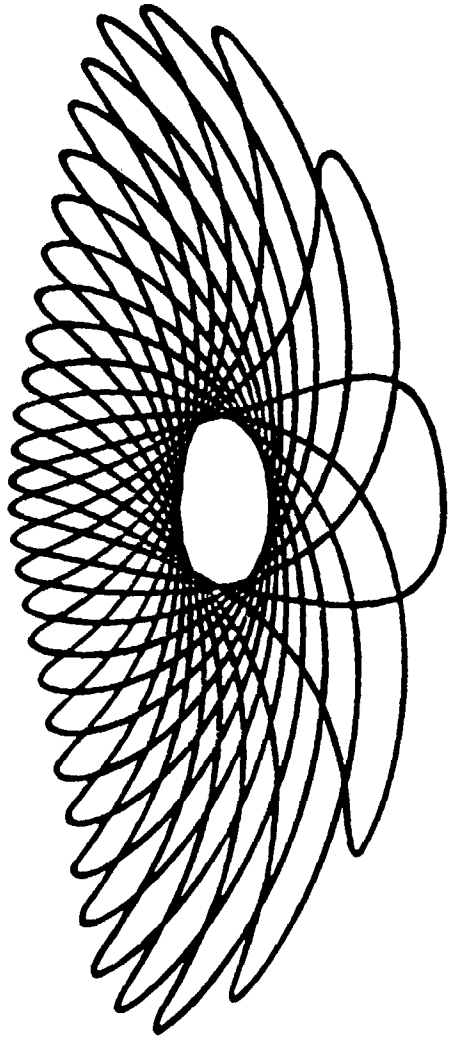
The designs below were made with ring 155/105.

Wheels



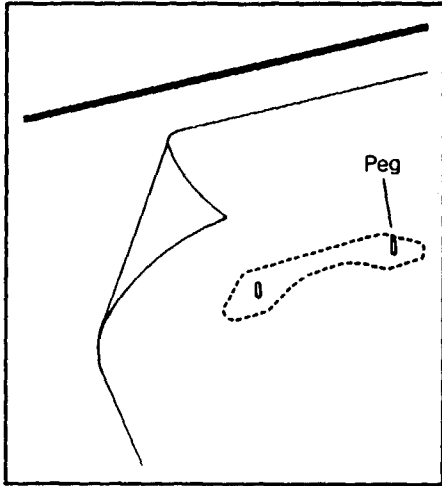
Appendix A: Current Instruction Booklet

How To Design With Spirograph Plus™ Design Toy

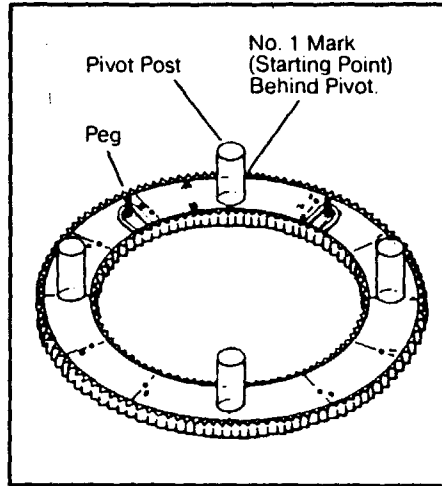


SPIROGRAPH
PLUS
DESIGN TOY ^{T.M.}

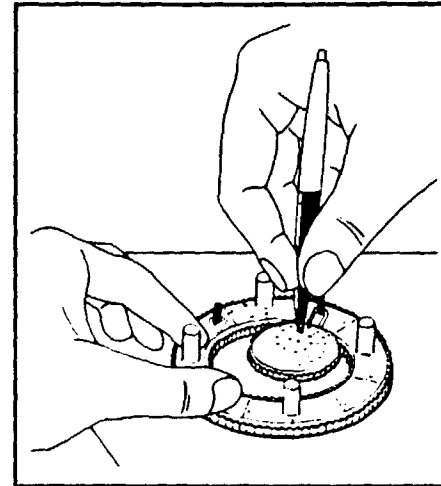
BASIC PATTERN INSTRUCTION



- The **Holder** is used to position **Rings** and **Racks** while you draw the designs.
- Lay the **Holder** under a sheet of paper.
- Press down on the paper to punch the **Pegs** thru it.



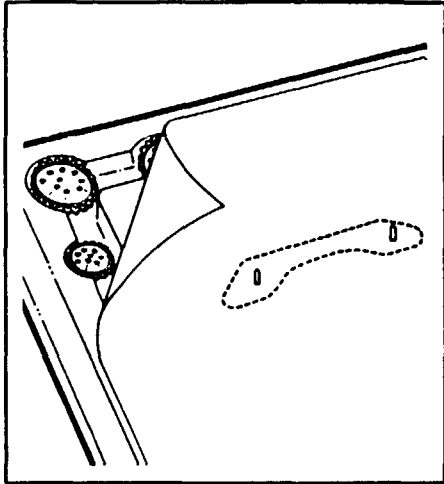
- Select a **Ring** and put the indicated **Holes** in the **Ring** onto the **Pegs**.
- Be sure the **Holder** does not interfere with your design.
- Each **Ring** is numbered.
- The Number near the outside edge indicates the number of outside **Gear Teeth**; the number near the inside of the **Ring** indicates the number of inside **Gear Teeth**.



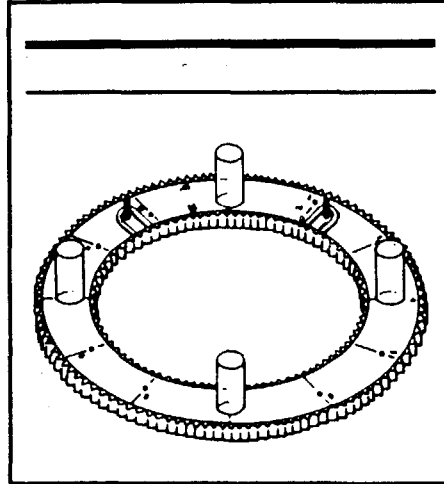
- Select a **Wheel** or **Gear** and place it inside the **Ring**.
- Each **Wheel** or **Gear** is numbered as well as having a Number near each **Hole** to help you locate the proper **Ring** and **Hole** to make the desired design.

- **Note** ● Before doing your design, rub the point of the **Ball Point Pen** on a piece of scratch paper until the ink flows smoothly.
- Put a **Ball Point Pen** (only) into one of the numbered **Holes** of the **Wheel**. With one hand hold the **Ring** down. With the other hand, hold the **Ball Point Pen** upright, do not press too hard, and carefully move the **Wheel** around the inside of the **Ring**.
- Always keep the **Teeth** of the **Wheel** and **Ring** in contact with each other while drawing your designs.
- Start at **Hole** number 1 on the **Ring** and continue until your line meets where you started.

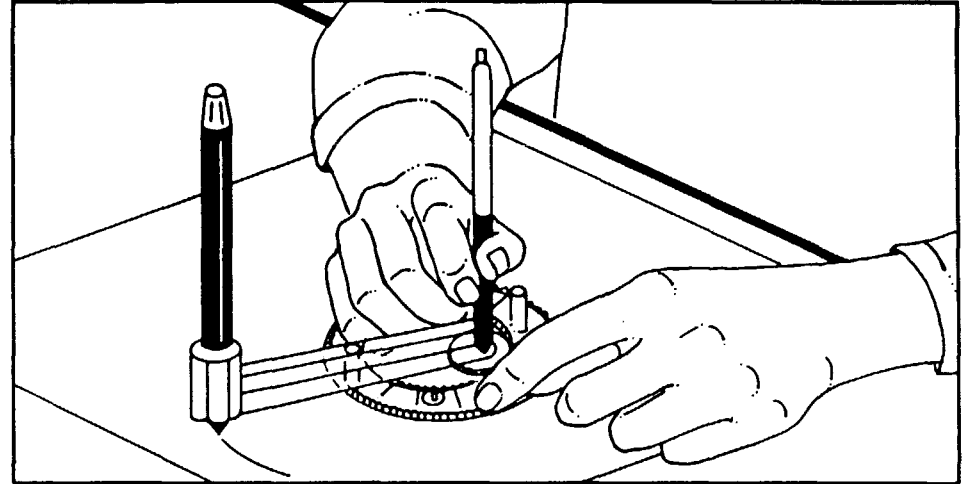
CREATING DESIGNS WITH THE GYRO ARM



- Turn **Holder** over in the tray so Peg side is up.
- Line Paper up with top, bottom and left side of Tray.
- Press Paper down on **Holder** punching Pegs thru the Paper.
- Pick the Paper and the **Holder** up and place them on a hard, flat surface for drawing.

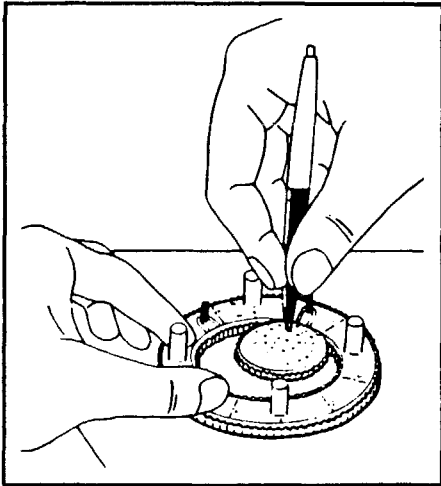


- Select a **Ring** and put the indicated Holes in the **Ring** onto the Pegs of the **Holder**.
- Select a **Wheel** and place it inside the **Ring**.
- Place a **Fiber Tipped Pen** (only) point end into the tube end of the **Gyro Arm** and push down on the **Pen** until it is secure in the Tube.

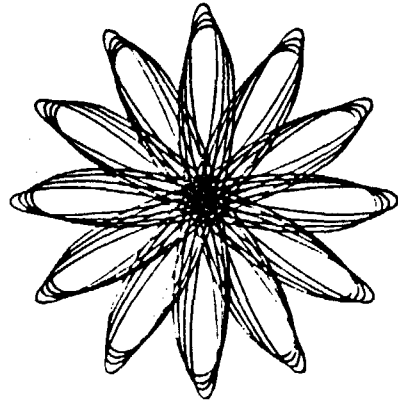


- This allows the correct amount of the point, (almost $\frac{1}{4}$ ") to show beyond the bottom of the **Gyro Arm** Tube.
- Slide the **Gyro Arm** onto a Pivot Post on the **Ring**, line up the Hole in the opposite end of the **Gyro Arm** with the desired Hole in the **Wheel** or **Gear**.
- Select a **Ball Point Pen** and place it, point end thru the Hole in the **Gyro Arm** and the selected Hole in the **Wheel** or **Gear**.
- Proceed with your design holding the **Ring** steady with one hand and moving the **Wheel** or **Gear** with the **Ball Point Pen** as though you were doing any basic design, but more *slowly*.
- The **Gyro Arm** will move with your motion creating a Skewed Design on the paper.

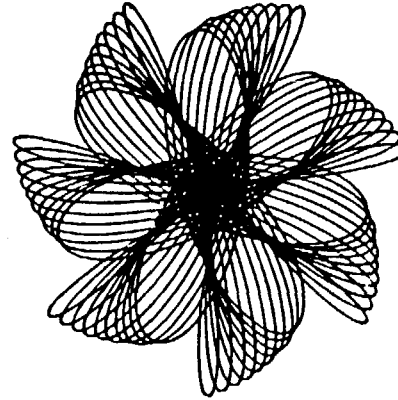
DESIGNS DRAWN WITH WHEELS OR GEARS INSIDE RINGS



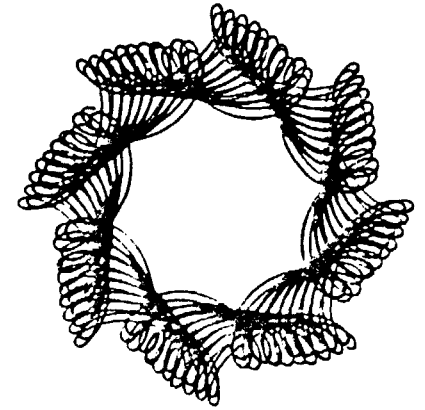
- Place **Holder** at desired location on paper. Push Pegs thru.
- Put paper with **Holder** on smooth hard surface to begin drawing.
- Place **Ring** on **Holder**. Use Pegs to position **Ring**.
- Select **Wheel**, place it inside **Ring**, and insert **Ball Point Pen**.
- Draw your design.



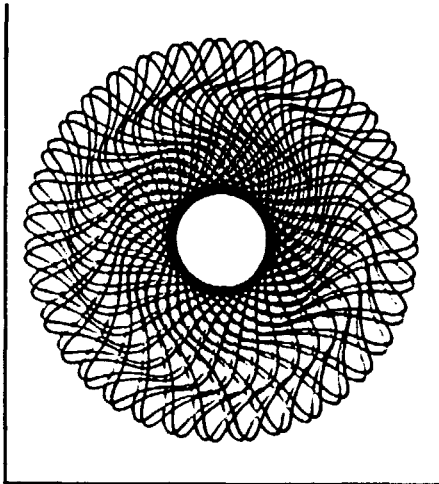
- Use **Ring No.** $\frac{144}{96}$
- Use **Wheel No 56**, line up Hole 1 with Mark 1 on **Ring**.
- With **Blue Pen** in Hole 1 draw the first pattern.
- Reposition **Wheel**, line up Hole 3 with Mark 1, draw the second pattern.
- Repeat using Holes 5 and 7.



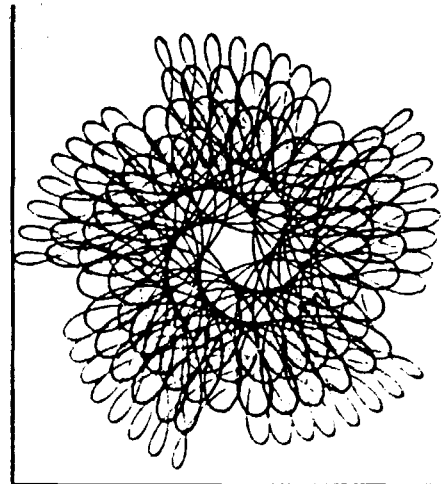
- Use **Ring No.** $\frac{150}{105}$
- Use **Wheel No. 60**, line up Hole 1 with Mark 1 on **Ring**.
- With **Red Pen** in Hole 1 draw the first pattern.
- Reposition **Wheel**, line up Hole 2 one Tooth to the right (of the 1st pattern) draw second pattern.
- Repeat, using Holes 3 thru 10.



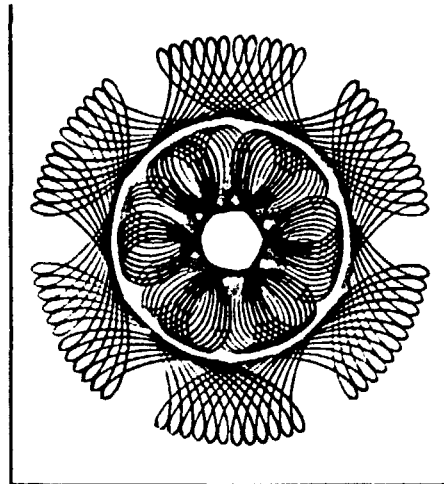
- Use **Ring No.** $\frac{144}{96}$
- Use **Wheel No. 84**, line up Hole 2 with Mark 1 on **Ring**.
- With **Orange Pen** draw patterns using Holes 2 thru 10, moving 2 Teeth to the right for each pattern.
- Line up Hole 4 with Mark 1 and draw pattern using Holes 4 thru 12 moving 2 Teeth to the right for each pattern.



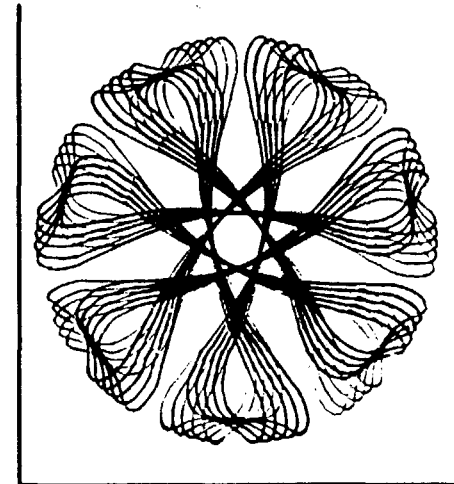
- Use Ring No. $\frac{150}{105}$
- Use Triangle Gear, line up Hole 12 with Mark No. 1 on Ring.
- With Orange Pen in Hole 12 draw the first pattern.
- Move Gear one Tooth to the right and
- Using Red Pen in Hole 12, complete the design.



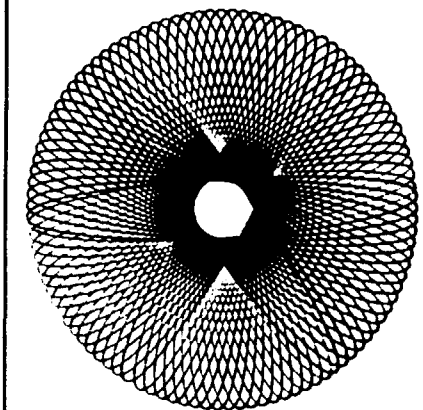
- Use Ring No. $\frac{150}{105}$
- Use Wheel No. 84, line up Hole 1 with Mark 1, with Blue Pen draw first pattern.
- Move Wheel two Teeth to the right. Using Hole 2, draw pattern.
- Move Wheel one Tooth to the right. Using Holes 3 thru 8, draw patterns.
- Use Red Pen. Move Wheel two Teeth to the right, use Hole 9, draw pattern.
- Move Wheel one Tooth to the right. Using Holes 10 thru 22, draw patterns.



- Use Ring No. $\frac{144}{96}$
- Use Wheel No. 80, line up Hole 1 with Mark 1. Draw six Red patterns, then draw six Orange patterns moving one Tooth right each time.
- Line up Hole 20 with Mark 1. Draw six Orange patterns, then six Red patterns moving one Tooth right each time.

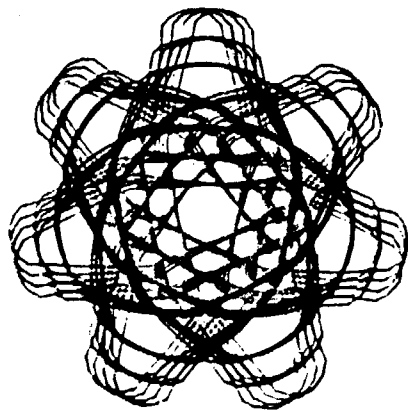


- Use Ring No. $\frac{150}{105}$
- Use Oval Gear, line up Hole 13 with Mark 1. Draw five Blue patterns moving one Tooth right each time.
- Turn Gear over (numbers face down). Draw five Red patterns moving one Tooth left each time.



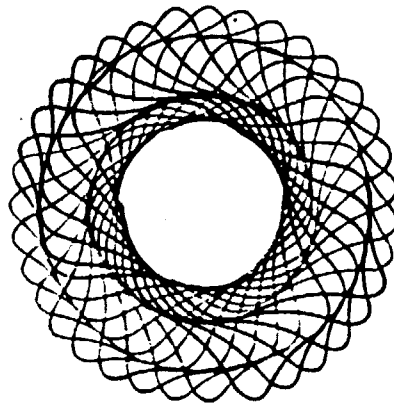
● Use Ring No. $\frac{150}{105}$

- Use **Wheel No. 52**, line up Hole 1 with Mark No. 1. With **Blue Pen** draw eighteen loops.
- Without moving the **Wheel** change to **Red Pen** and draw eighteen more loops.
- Change to **Orange Pen** and complete design.



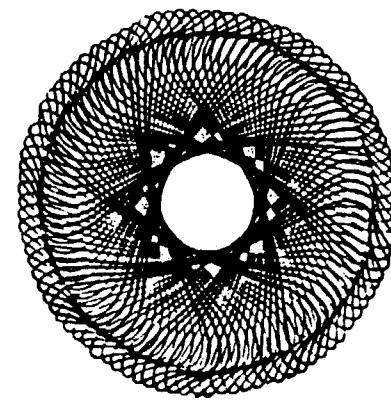
● Use Ring No. $\frac{150}{105}$

- Use **Quad Gear**. Line up Hole 4 with Mark 1; with **Red Pen** draw four patterns, moving one Tooth each time.
- Use **Blue Pen**. Line up Hole 3 with Mark 1, draw 4 patterns moving one Tooth right each time.
- Use **Orange Pen**. Line up Hole 2 with Mark 1, repeat as above making five designs.



● Use Ring No. $\frac{150}{105}$

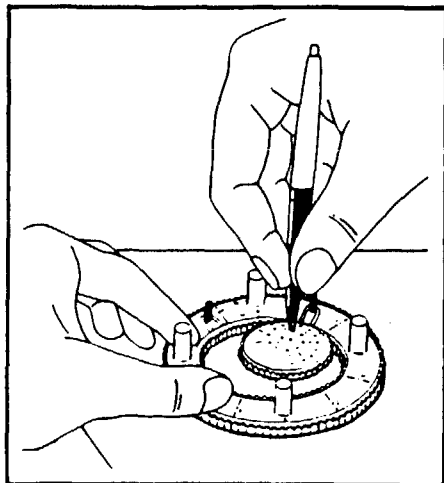
- Use **Triangle Gear**, line up Hole 9 with Mark 1; with **Red Pen** draw 6 loops.
- Without moving the **Gear** change to **Blue Pen** and draw six more loops.
- Change to **Orange Pen** to complete design.



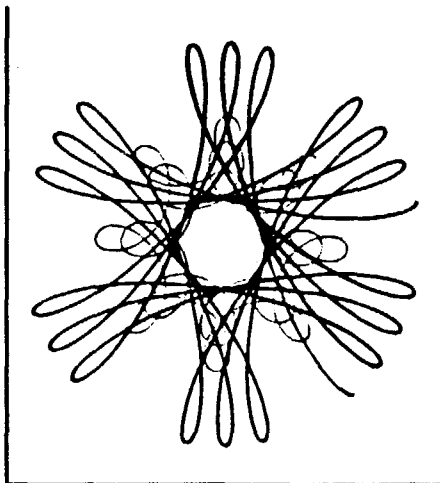
● Use Ring No. $\frac{144}{96}$

- Use **Oval Gear**, line up Hole 13 with Mark 1, with **Blue Pen** draw three patterns moving one Tooth right each time.
- Continue as above using **Orange Pen**, draw five more designs.
- Continue using **Red Pen**, draw four more patterns to complete design.

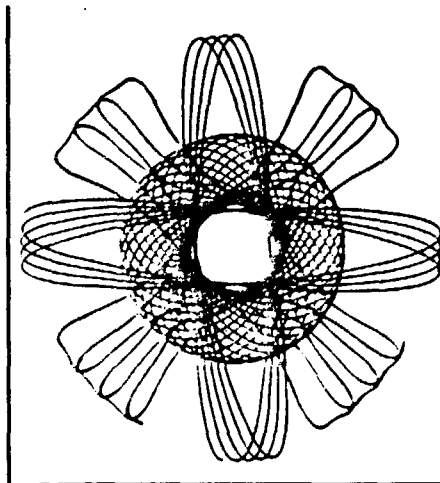
PARTIAL DESIGNS DRAWN WITH WHEELS OR GEARS INSIDE RINGS.



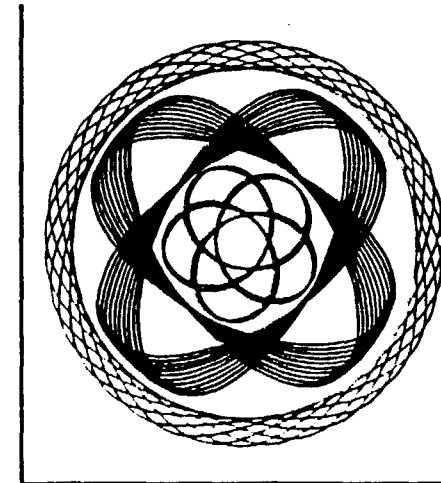
- Place **Holder** at desired location on paper. Push Pegs thru.
- Put paper with **Holder** on smooth hard surface to begin drawing.
- Place **Ring** on **Holder**. Use Pegs to position **Ring**.
- Select **Wheel**, place it outside **Ring** and insert **Ball Point Pen**.
- Draw your design.



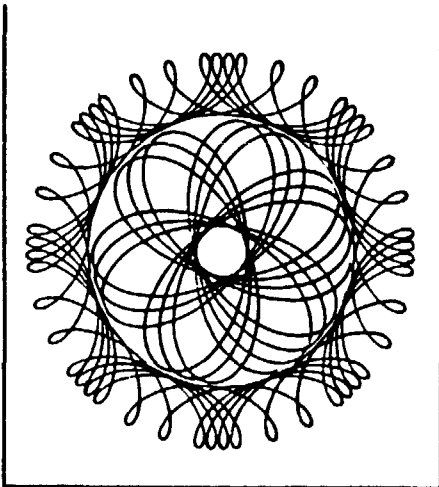
- Use **Ring No.** $\frac{144}{96}$
- Use **Wheel No. 63**, line up Hole 1 with Mark 1 on **Ring** with **Red Pen** make only 3 loops.
- Line Hole 1 with Mark 11 and make 3 more loops.
- Use **Wheel No. 84**, line up Holes 17 with Mark 1 make pattern with **Blue Pen**.
- Line up Hole 22 with Mark 1 and make pattern with **Blue Pen**.



- Use **Ring No.** $\frac{150}{105}$
- Use **Wheel No. 52**. Line up Hole 2 with Mark 1 on **Ring**. With **Blue Pen** make 4 loops. Move **Wheel** $\frac{1}{4}$ way around **Ring** and draw 4 more loops.
- Use **Triangle Gear**. Line up Hole 1 with Mark 2 on **Ring**. Make 3 loops. Move **Wheel** $\frac{1}{4}$ way around **Ring**, draw 3 loops.
- Use **Wheel No. 72**, Hole 24 to complete pattern.



- Use **Ring No.** $\frac{150}{105}$
- Use **Wheel No. 84**, line up Hole 32 with Mark 1. With **Orange Pen** make two patterns moving two Teeth right between them.
- With **Wheel No. 52**, Hole 13. With **Red Pen** make ten loops. Move wheel $\frac{1}{4}$ way around **Ring** and make ten more loops.
- Use **Wheel No. 24**, Hole 4. Use **Blue Pen** to finish.

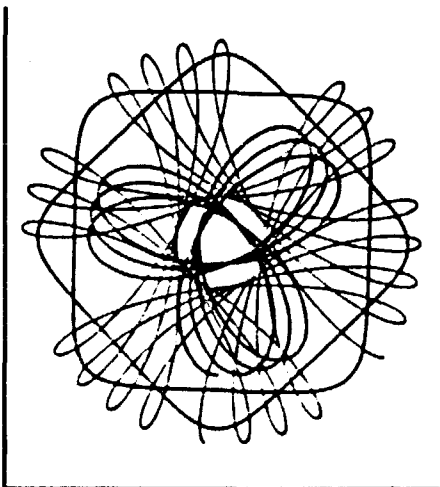


● Use Ring No. $\frac{144}{96}$

● Use **Wheel No. 84**. Line up Hole 1 with Mark 1 on **Ring**. With **Red Pen** draw 4 patterns, moving **Wheel** 1 Tooth right each time.

● Move **Wheel** 3 Teeth right. Draw 1 pattern. Move **Wheel** 3 Teeth right, repeat pattern.

● Use **Wheel 63**, line up Hole 16 with Mark 1 on **Ring**. Draw 3 loops. Line up Hole 16 with Mark 11 on **Ring**. Draw 3 loops.

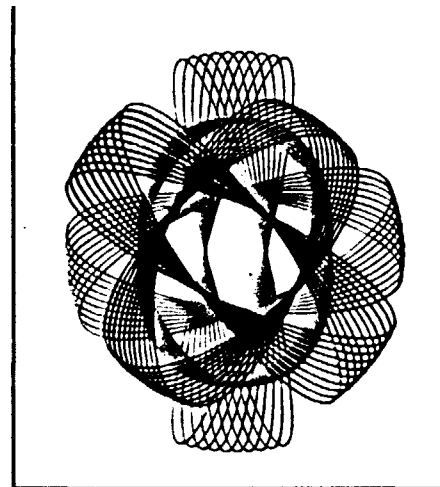


● Use Ring No. $\frac{144}{96}$

● Use **Wheel No. 63**, line up Hole 1 with Mark 1 on **Ring**. With **Blue Pen** draw three loops. Line up Hole 1 with Mark 11. Draw four more loops.

● Line up Hole 15 with Mark 1. With **Crange Pen** draw three loops.

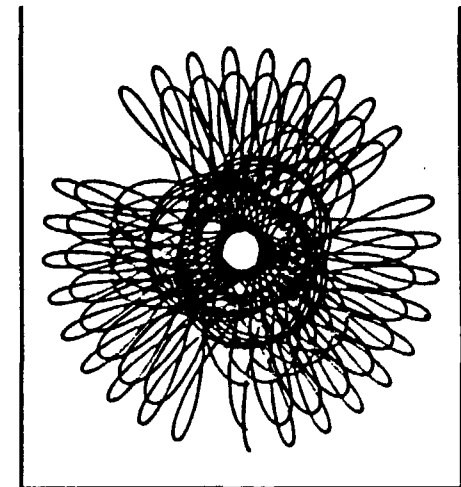
● Use **Wheel No. 24**, Hole 4. With **Orange Pen** make two evenly spaced patterns.



● Use Ring No. $\frac{150}{105}$

● Use **Wheel No. 52**, line up Hole 1 with Mark 1 on **Ring**. Using **Blue Pen** draw ten loops. ● Move **Wheel** five Teeth right. Use **Red Pen**, in Hole 6. Draw ten loops.

● Move **Wheel** fifteen Teeth right and continue design alternating **Blue** and **Red Pens** in Holes 9, 12, and 18 to complete design.



● Use Ring No. $\frac{144}{96}$

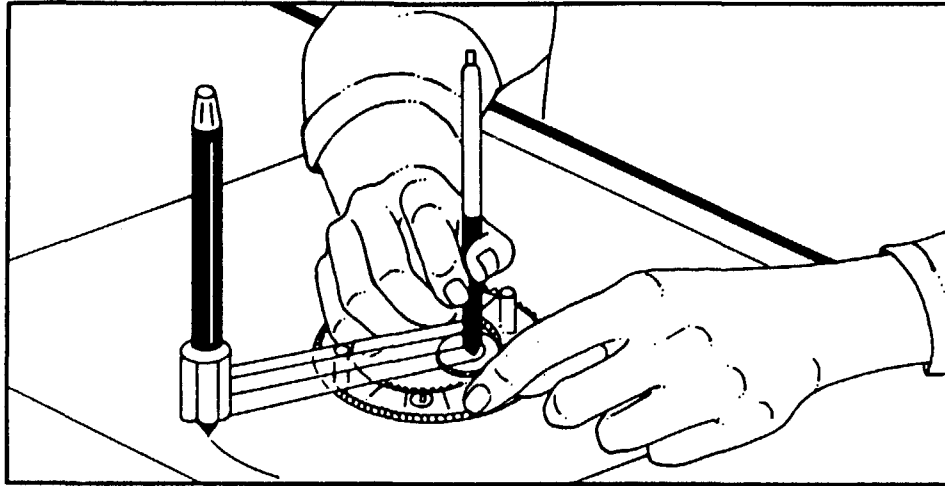
● Use **Wheel No. 63**, line up Hole 1 with Mark 1 on **Ring**. With **Orange Pen** draw 4 loops; move to mark 3 and draw 4 more loops.

● Line up Hole 6 with Mark 1 on **Ring**. With **Red Pen** draw 8 loops.

● Line up Hole 15 with Mark 1. With **Orange Pen** draw 4 loops.

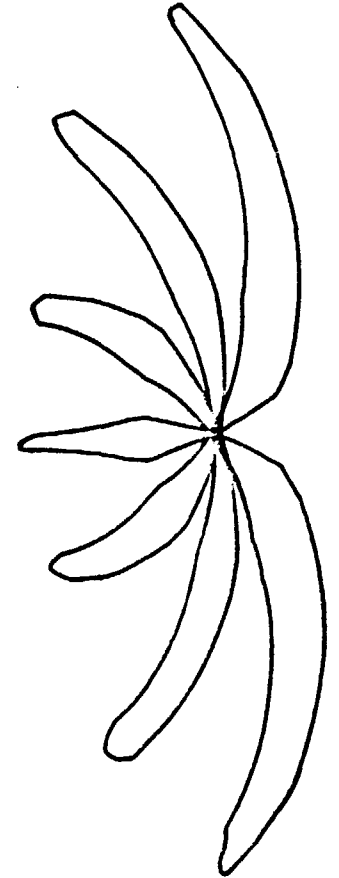
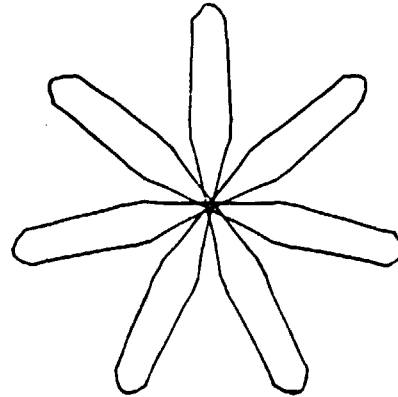
● Line up Hole 24 with Mark 1. With **Red Pen** draw 4 loops.

SKewed DESIGNS DRAWN WITH WHEELS OR GEARS USING GYRO ARM INSIDE RINGS



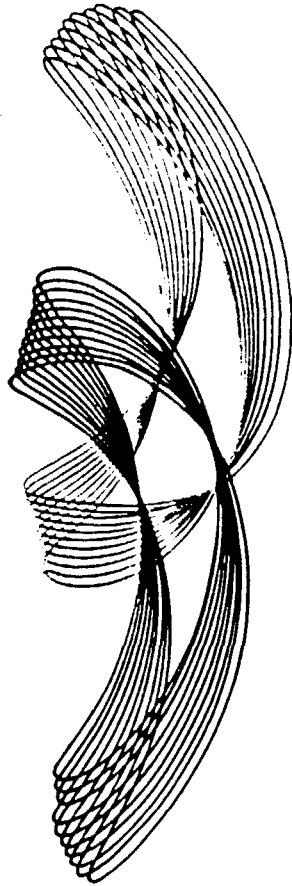
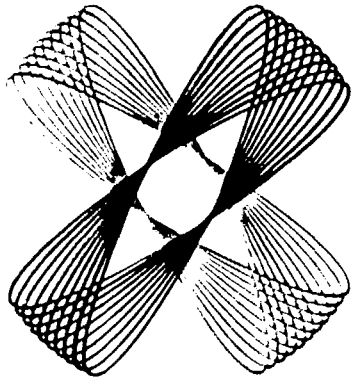
● Refer to basic instructions for Creating Designs with the **Gyro Arm** on page 2.

- Use Ring No. $\frac{150}{105}$
- Use Quad Gear Hole 1. Using Red Pen in Wheel and Blue Fiber Tipped Pen in Gyro Arm draw one pattern.



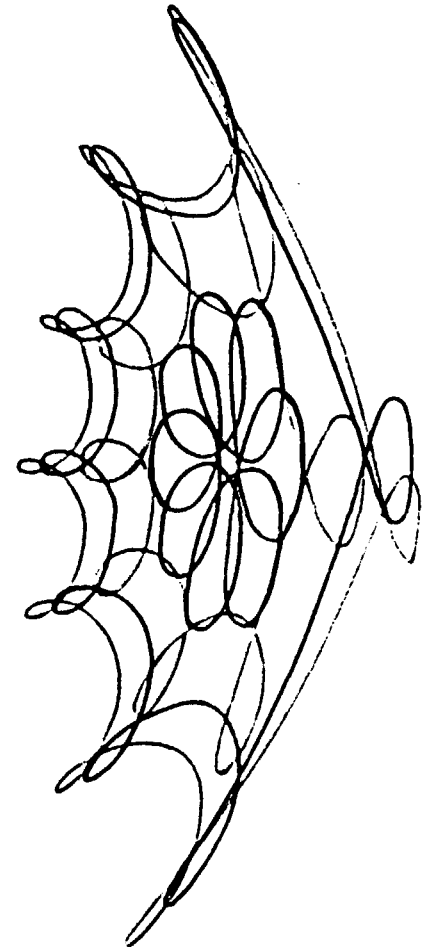
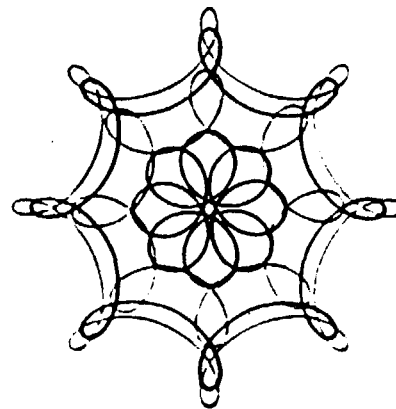
● Use **Ring No.** $\frac{150}{105}$

- Use **Wheel No. 52**, line up Hole 3 with Mark 1. With **Blue Pens** make ten loops.
- Switch to **Red Pens**, move halfway around **Ring** and draw ten loops.



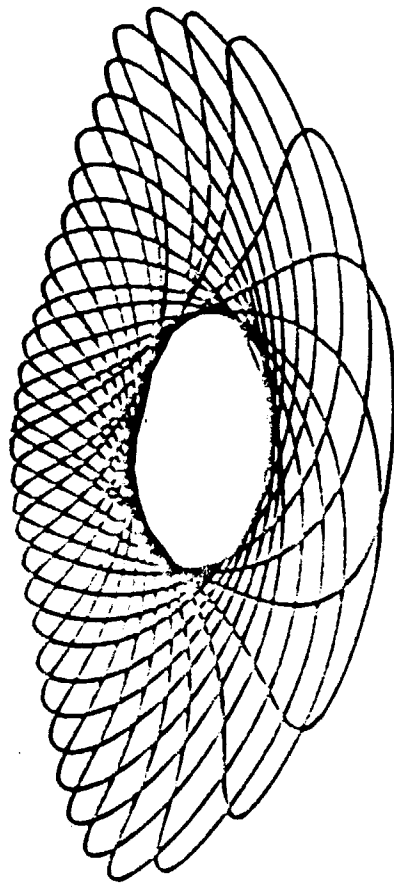
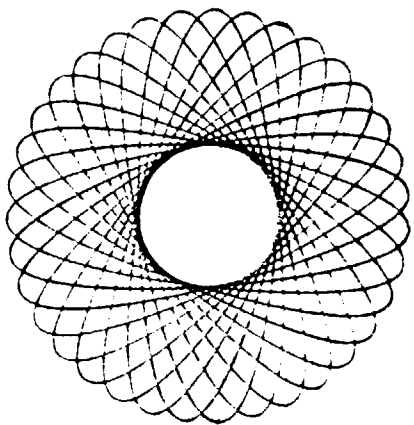
● Use **Ring No.** $\frac{144}{96}$

- Use **Wheel No. 84**, line up all Holes with Mark 1 on **Ring**.
- Draw patterns with Holes 6, 10, 18, and 30 alternating **Blue** and **Red Pens**



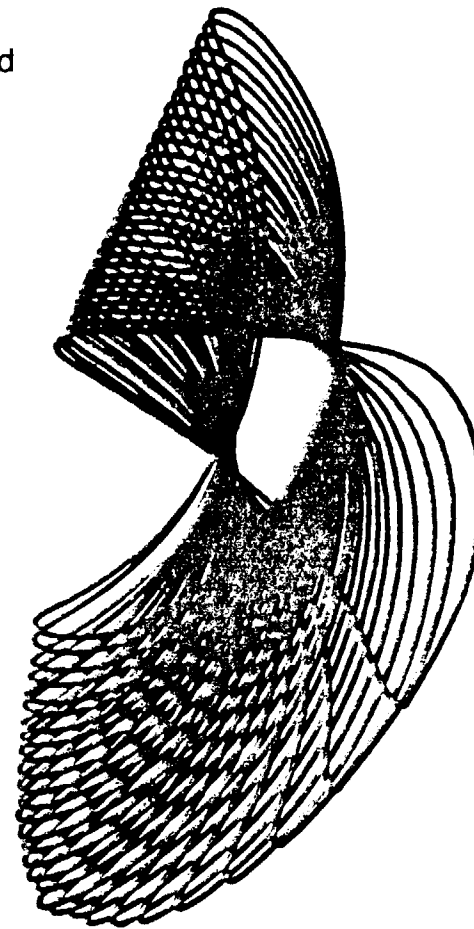
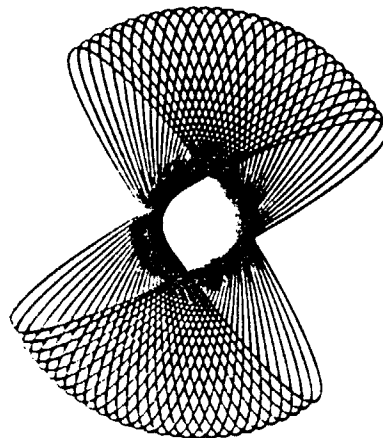
● Use Ring No. $\frac{150}{105}$

● Use Wheel No. 48, line up Hole 6 with Mark 1. Use Blue Pens to draw designs.

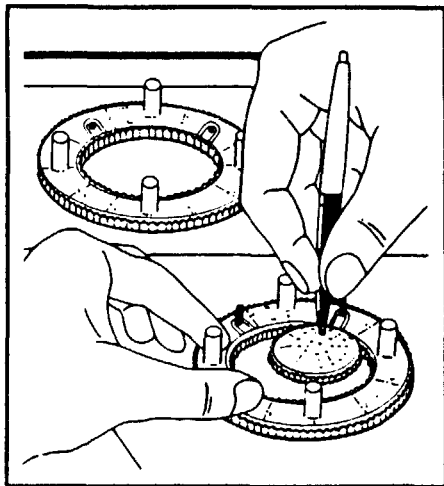


● Use Ring No. $\frac{150}{105}$

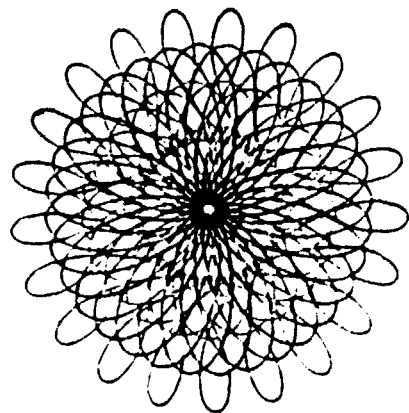
● Use Wheel No. 52, line up Hole 3 with Mark 12 on Ring. With Blue Pen in Gyro Arm, and Orange Pen in Wheel draw partial design (24 loops)



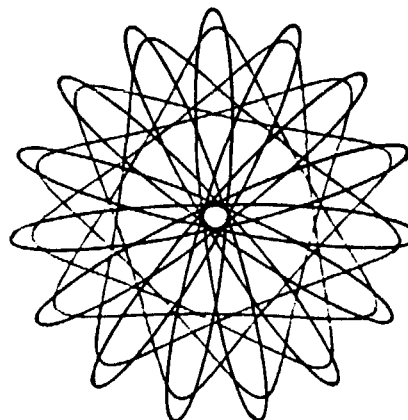
DESIGNS DRAWN USING BOTH RINGS



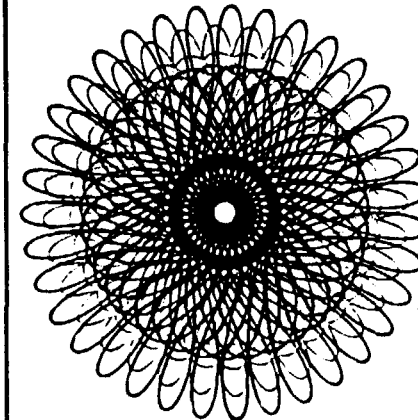
- Place **Holder** at desired location on paper. Push Pegs thru.
- Put paper with **Holder** on smooth hard surface to begin drawing.
- Place first **Ring** on **Holder**. Use Pegs to align.
- Select **Wheel** or **Gear** and insert **Ball Point Pen**.
- Draw your design.
- Continue this procedure with other **Ring**.



- Use **Rings No. $\frac{150}{105}$ and $\frac{144}{96}$**
- **Ring No. $\frac{150}{105}$**
- Use **Wheel No. 80**, line up Hole 9 with Mark 1. With **Blue Pen**, draw pattern.
- Line up Hole 23 with Mark 1, draw pattern.
- **Ring No. $\frac{144}{96}$**
- Use **Wheel No. 63**, line up Hole 11 with Mark 1. Use **Orange Pen**, draw pattern.

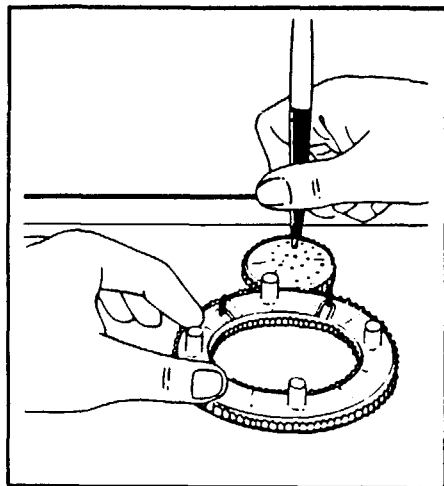


- Use **Rings No. $\frac{150}{105}$ and $\frac{144}{96}$**
- **Ring No. $\frac{150}{105}$**
- Use **Wheel No. 56**, line up Hole 1 with Mark 1. With **Red Pen**, draw pattern.
- **Ring No. $\frac{144}{96}$**
- Use **Wheel No. 32**, Hole 2. With **Orange Pen** draw five patterns starting at a different point of the **Red** pattern each time.



- Use **Rings No. $\frac{150}{105}$ and $\frac{144}{96}$**
- **Ring No. $\frac{150}{105}$**
- Use **Wheel No. 72**, with **Red Pen** draw patterns using Holes 5 and 17, lining up the Holes with Mark 1 on the **Ring** each time.
- **Ring No. $\frac{144}{96}$**
- Using **Wheel No. 63**, draw patterns using Holes 5 and 10 with the **Blue Pen** in the same way as above.

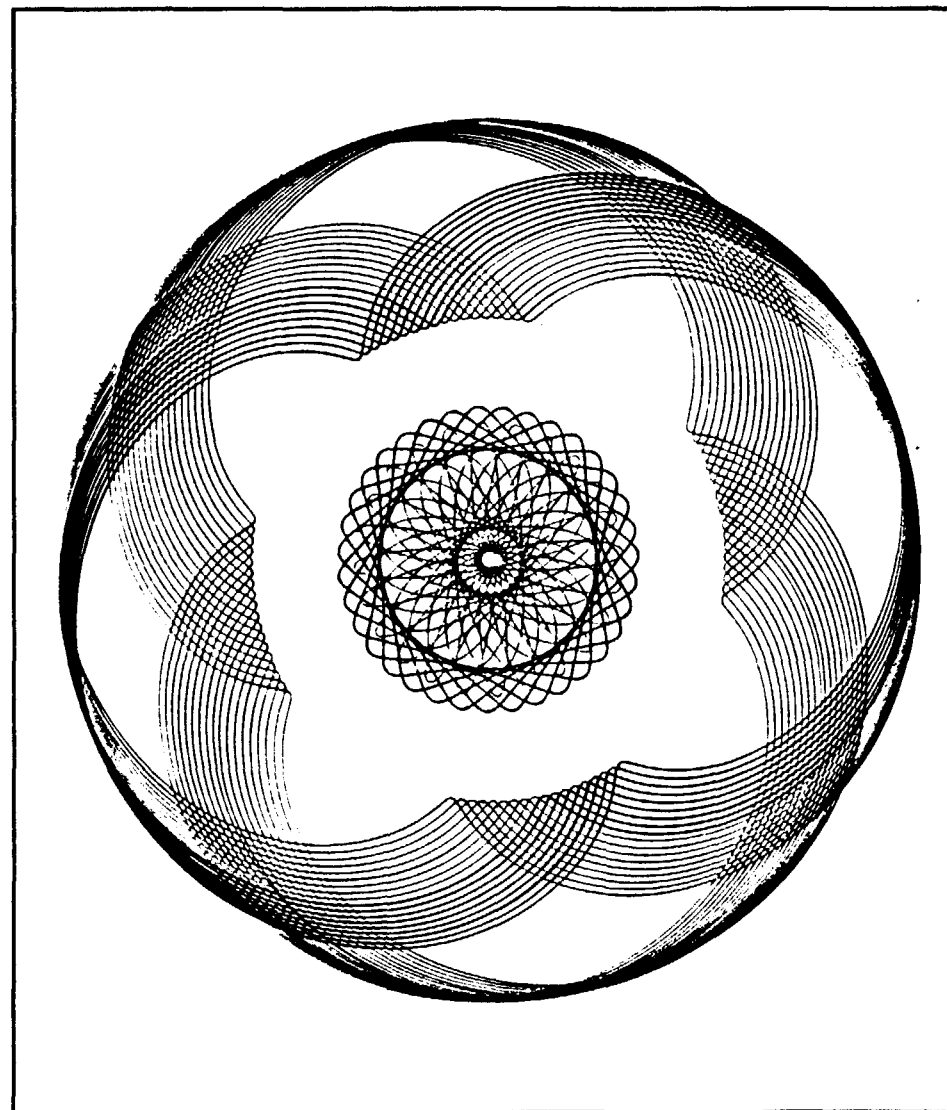
DESIGNS DRAWN USING WHEELS OUTSIDE OF RINGS



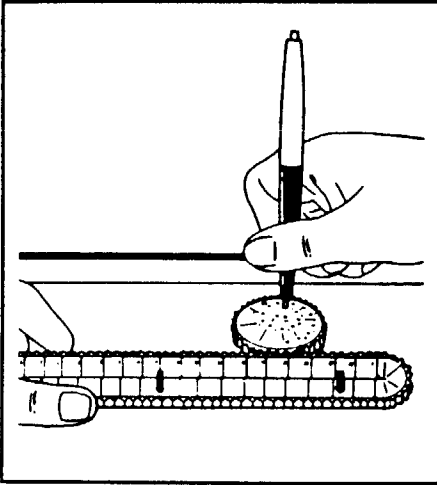
- Use **Ring No. $\frac{144}{96}$**
- Use **Wheel No. 72**, line up Hole 1 with Mark 1 on outside edge **Ring**.
- Use **Blue Pen** draw nine patterns moving one Tooth right for each. Continue to draw nine **Orange** patterns.
- Line up Hole 1 with Mark 4 of the **Ring** and draw nine **Orange** patterns.

- **Inside Design.**
- Leave **Ring** in place.
- Use **Wheel No. 52**, Hole 1, draw one **Orange** pattern. Use Hole 10, draw one **Blue** pattern.
- Remove **Ring No. $\frac{144}{96}$**
- Use **Ring No. $\frac{150}{105}$**
- Use **Wheel No. 24**, Hole 1, draw one **Red** pattern.

- Place **Holder** at desired location on paper. Push **Pegs** thru.
- Put paper with **Holder** on smooth hard surface to begin drawing.
- Place **Ring** on **Holder**. Use **Pegs** to position **Ring**.
- Select **Wheel**, place it inside **Ring** and insert **Ball Point Pen**.
- Draw your design.



DESIGNS DRAWN WITH WHEELS AROUND RACK



- Place **Holder** in center of paper. Push Pegs thru.
- Put paper with **Holder** on smooth hard surface to begin drawing.
- Position **Rack** on **Holder** Pegs.
- Select **Wheel**, place on outer edge of **Rack** and insert **Ball Point Pen** into desired Hole in **Wheel**.
- Draw your design.

- Use **Wheel No. 56**
- Line up Hole 6 with Mark 11 on the **Rack**. With **Red Pen** draw four loops. Change to **Orange Pen** and draw two loops.
- Change to **Blue Pen** and draw two more loops.
- Complete design using **Orange Pen**

- For additional designs move the **Rack** from side to side using different Holes.

