HARRY STEWART

ATARI WSFN

AN INTRODUCTION....

ATARI WSFN INTRODUCTORY MANUAL

Prepared by: Gregory Yob

PREFACE (General remarks that are unlikely to appear in the final version of this manual.)

1) ATARI WSFN is subject to changes which may require changes to this manual- for example, the representation of the turtle might change.

2) It is assumed that the user is familiar with the ATARI keyboard, and knows how to manipulate the SHIFT and CTRL keys.

3) Entry of keys will be as typed with these exceptions:

Letter ^{C} means CTRL-Letter

Letter ^{S} means SHIFT-Letter

For example, $M^{C}2T^{C}1HCN$ is equivalent to:

CTRL-M 2 CTRL-T 1 H C N

WHAT'S WSFN ????

Have you ever tried to build a house out of toothpicks? Or to bail out a battleship with a teaspoon? That's the situation that the first computer hobbyists were faced with.....

Back in the "old days" of personal computers, you would buy some kits through the mail, and with a soldering iron, a lot of patience, and even more luck, you would end up with your very own personal computer! However, that was only half the story - for a computer without a program is like a car out of gas - or your ATARI 400 (or ATARI 800) without any cartridges. You could turn your computer on - and it would sit there and do nothing.....

Deep inside every computer is the electronic equivalent of a blackboard, and written on the blackboard is a series of instructions for the computer to follow. You might imagine an ordinary blackboard with a gridwork of 1/2 inch squares on it - if you filled some squares with chalk, and erased others, you can draw letters - and then words and sentences. Of course it takes hundreds of little squares to write a short sentence.....

When a computer is turned on, it looks at its memory (that's the blackboard), and the pattern of ones and zeroes (white or black squares) contains the instructions that tell it to do something. Since it is easier and cheaper to build computers that recognize "binary patterns" that to make computers that understand English sentences, the computer's memory will look very strange to you and me.

Now, back in the "old days", the people with big computers had already solved this problem - patterns of ones and zeroes were made which let humans use typewriters and English-like sentences to tell the computer what to do. (This is called programming.) But!! The poor fellow with his personal computer didn't have these computer languages - and his personal computer's memory was very small. In fact, the memory was too small to hold the languages that big computers were used to.

Computer hobbyists soon got tired of flipping switches thousands of times to make their programs - so "tiny" languages were invented. A tiny language couldn't do as much as the languages on big comuters, but you can still do a lot more in a tiny language, and do it a lot faster, than making up the ones and zeroes by hand.

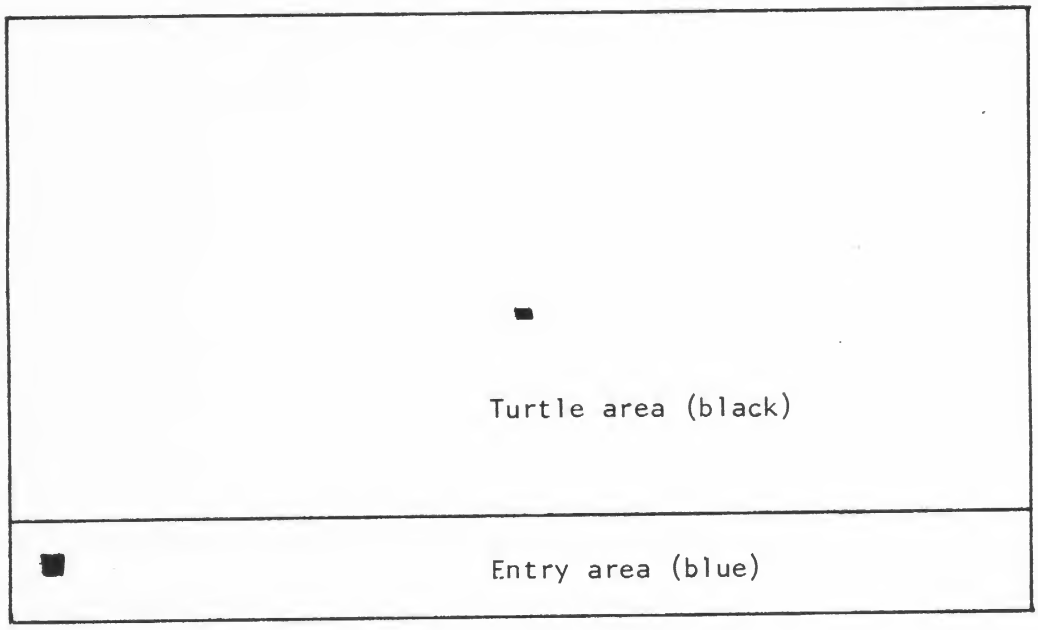
In 1977, Lichen Wang, who had already made a Tiny BASIC, decided to make a tiny language for controlling robots with his computer. When he shared it with his friends, they started to call it different things - since Lichen hadn't given the language a name. When Lichen was asked what to call the language, he said: "If everything has to have an acronym, I would rather call it WSFN (Which Stands For Nothing)."

When Lichen had WSFN ready, he didn't have a robot - but he did have a TV set, and WSFN could be made to draw pictures with the Turtle. (You can think of the Turtle as a video robot if you want to.) Well, lets plug in our ATARI 400 (or ATARI 800), put in the WSFN cartridge, and see what turtles can do.

THE TURTLE

Getting Started

Plug in the WSFN cartridge, and turn on the TV and your ATARI 400 (or ATARI 800). The TV should look like this:



WSFN when the power is turned on.

The top part of the screen will be black, with a bright dot in the center. The bottom part will be light blue, with a bright square near the left side.

The top half is the world of the turtle, and here is where pictures can be made. To draw pictures, you must enter commands with the keyboard, and the blue area will show your last entry and the line you are entering. (Nothing is shown when you start since nothing has been entered.)

How to See the Turtle

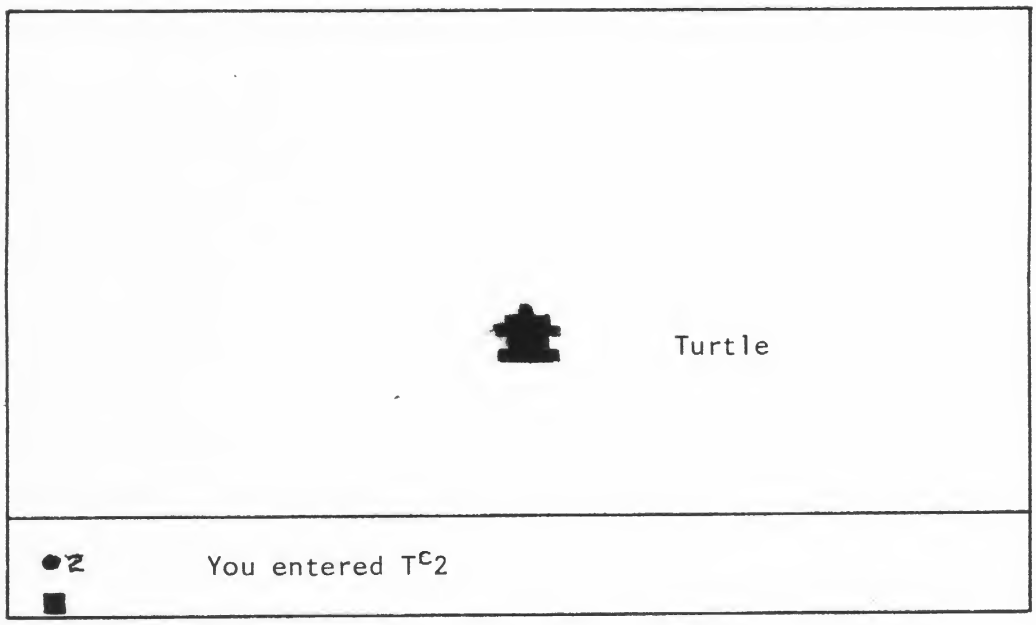
When WSFN starts up, the turtle is the bright dot - which doesn't look like a turtle! To see a better turtle, enter on the keyboard:

CTRL-T 2

Press the CTRL key, and while it is down, press the T key. Then press the 2 key (after releasing the CTRL key.)

NOTE: Instead of spelling SHIFT or CTRL for keystrokes, we will use s or c like this: M^{s} V^{c} - these mean SHIFT-M and CTRL-V. The turtle command shown above becomes T^{c} 2 .

Now the screen will look like this:



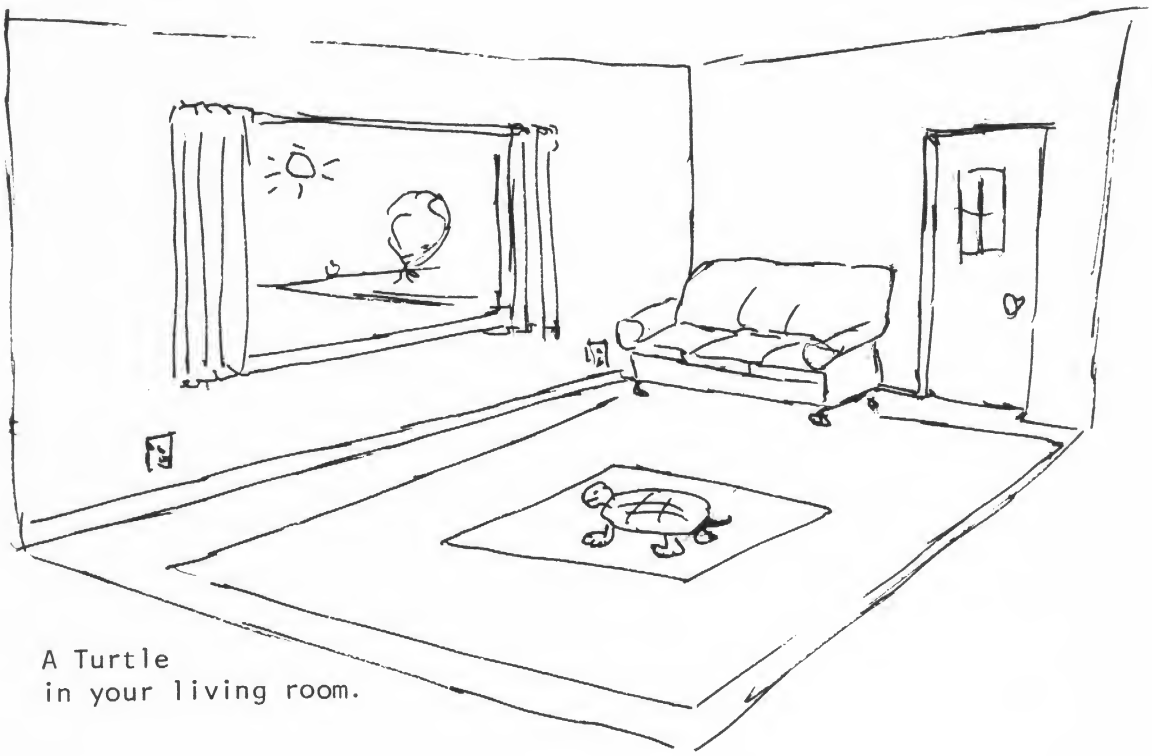
Making the Turtle appear.

The little animal on the screen is the turtle. If you look in the blue area, you will see a ● 2. The T^c is a graphics character, ●, so this represents the command you just typed in.

(If you have used a computer before, an unusual thing is going on here: WSFN DOES A COMMAND AS SOON AS IT RECOGNIZES IT. You don't have to press RETURN - WSFN is awares of what you type in as each key is pressed.)

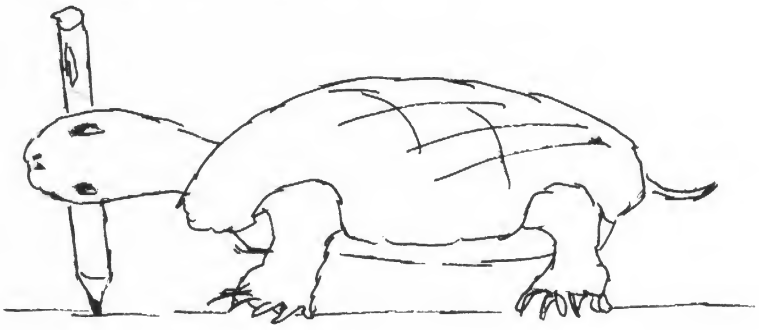
All About Turtles

Suppose you had a trained turtle, and you put him in the living room on a piece of paper:

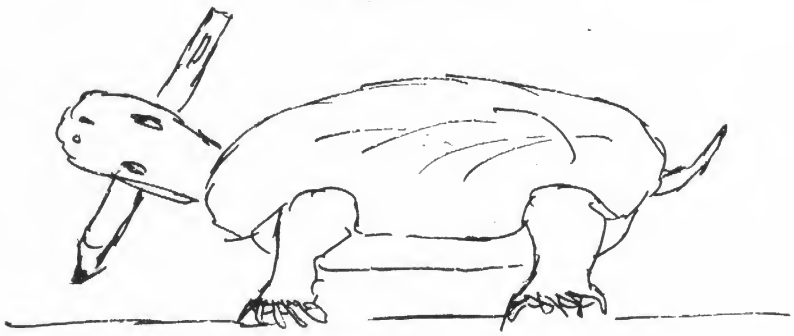


A Turtle
in your living room.

The turtle holds a felt pen in its mouth like this:

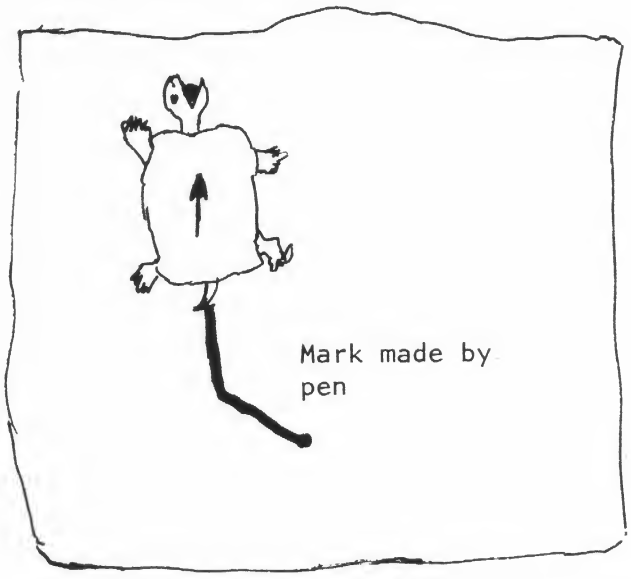


The pen is down.



The pen is up.

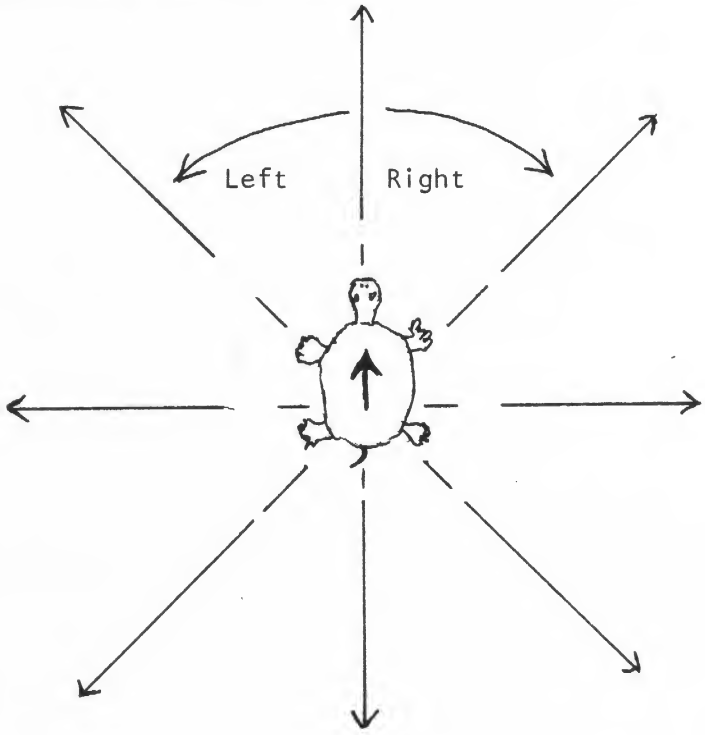
If the turtle walked forward, the felt pen would drag along the floor, leaving a mark on the paper:



Turtle goes for a walk with the pen down.

No mark would be made if the turtle was told to lift its head first.

Since turtles aren't too smart, you can only make him turn to the right or the left by half- turns like this:



How a Turtle changes directions.

Sometimes the turtle might walk off the paper - and you'll have to pick him up and put him back in the middle. If you want, you can tell the turtle to point towards the living room window.....

And, of course, when a drawing is finished, it's time to put a clean sheet of paper on the floor.

The WSFN turtle can do all of these things. The commands are:

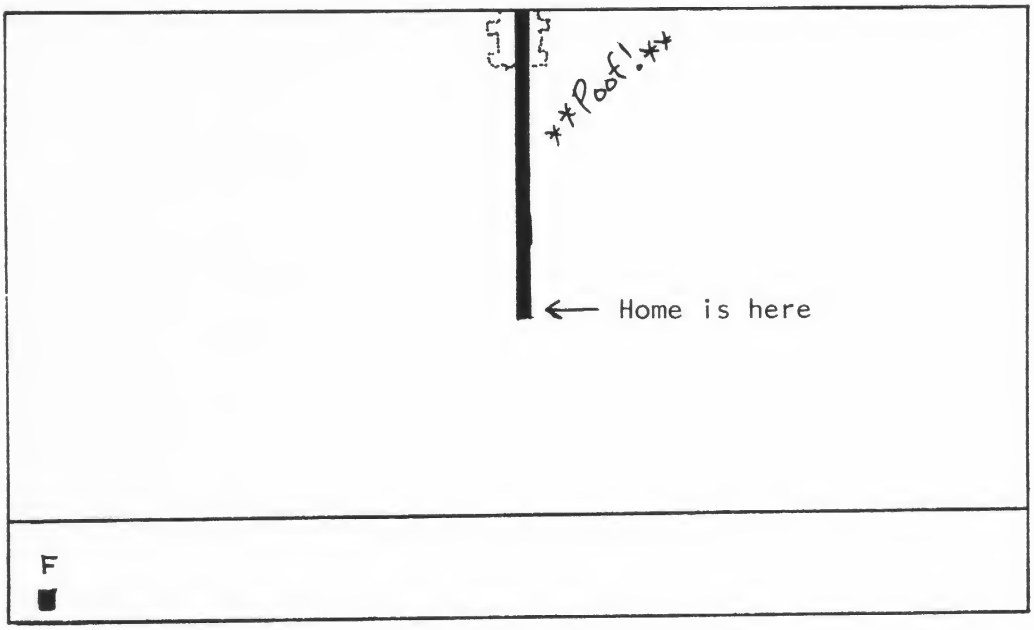
U - UP	Lift the pen off the floor.
D - DOWN	Put the pen on the floor.
F - FORWARD	Move one step forward.
R - RIGHT	Turn right one half-turn.
L - LEFT	Turn left one half-turn.
H - HOME	Put the turtle in the middle of the screen.
N - NORTH	Point the turtle North (up on the screen).
C - CLEAR	Clear the screen.

Using the WSFN Turtle

Press the F key (FORWARD) a few times - if you do it enough, the turtle will move upward and a line appears beneath him:

(Here you entered FFFFFFFFFFFFFF.)

If you hold the F key down for a while, the ATARI will repeat the key (with beeps for each repeat.). The turtle will travel up the screen and disappear off the top edge.



The Turtle walked off the top of the screen.

To get him back, use H (HOME) - and the turtle will reappear in the center again.

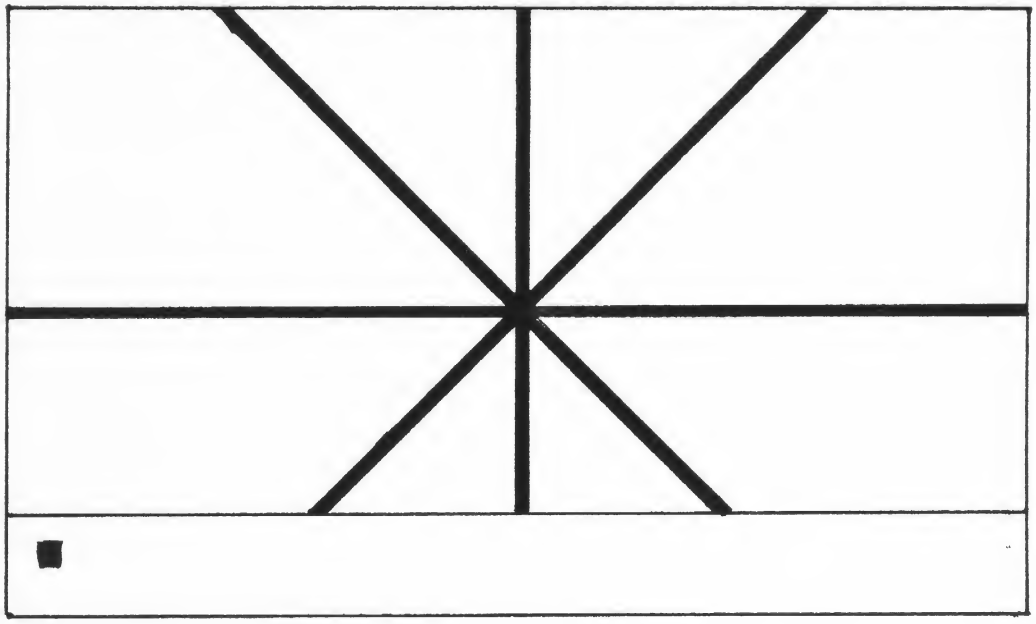
(Each time you press a turtle command key, WSFN does it immediately. This leaves only one letter as the last command shown in the bottom area on the screen.)

Now press R (RIGHT). The turtle will rotate a half-turn to the right. If you hold R down, the turtle will spin like a top! (Try it and see.)

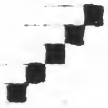
Pressing L (LEFT) turns the turtle to the left by a half-turn.

Here is a challenge for you: See if you can draw the picture shown below:

- ON NEXT PAGE -

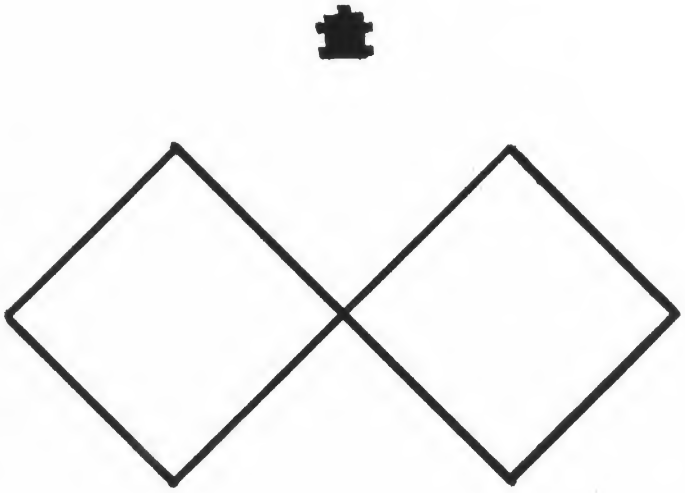


Can you draw this picture?

(Note: Diagonal lines will look like  on the screen.)

(When the turtle goes into the blue area, the turtle is visible, but the line it draws isn't shown.)

To erase the screen, press C (CLEAR). The picture goes away, with the turtle remaining where it was before. Clear the screen and see if you can draw this picture:



Another challenge for the drawing expert - and a small mystery.

How did the turtle get away from the drawing without a line?? Simple! He lifted his pen up - if you press U (UP), the turtle can be moved without making a line. Now see if you can do the drawing and leave the turtle in the spot shown.

(Hint: If you have a hard time making the lines meet in the center, just count the number of F's you entered - and then it's easy. Counting is very important in WSFN.)

To make the turtle draw again, use D (DOWN), and the pen will now make a line on the screen.

Make the turtle spin around by holding down the R key. Then press N (NORTH) and see what happens. Try this several times.

Some Optional Exercises

If you have a few moments, and want to really know your WSFN, try the exercises in this section. Some of them aren't easy, so feel free to skip them if you want to go on.

1) Draw a border around the edge of the screen. Keep count so you know how many "footsteps" high and wide the screen is.

2) Draw a square 15 steps on a side, and fill it in by walking around inside in a spiral.

3) Draw this figure:

Spend some time making pictures with the turtle, and then go on with the next section.

GETTING FANCIER

Suppose you wanted to draw a checkerboard on the screen. If you draw it one step at a time by pressing F, R and L, it will take a lot of effort! WSFN has several ways to save effort - so let's look at a few of these.

Repetition

If you enter a number before any WSFN command, WSFN will repeat the command. For example, clear the screen, home the turtle, point north, and enter:

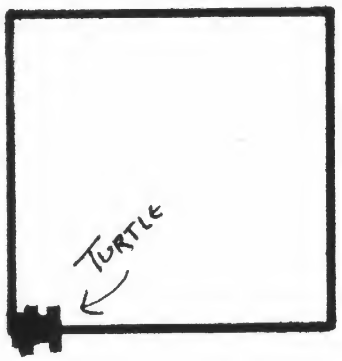
```
25F
```

The turtle will move 25 units forward. (Be sure not to put a blank between the 25 and the F. A blank is a WSFN command to do nothing, and if you enter 25(blank), WSFN will do nothing - 25 times!)

Making a square is easy! Try entering:

```
HCN25F2R25F2R25F2R25F
```

If you did that right, you will see:



To twirl the turtle, enter:

```
9999R
```

(NOTE: 9999 is the largest number that WSFN understands. If you enter a bigger number, WSFN will take the four rightmost digits as the repetition value.)

Parenthesis

When you drew the square, did you notice that a square could be made by repeating this command four times?

four of: 25F2R

In WSFN, groups of commands can be put inside pairs of parenthesis and then repeated just like one command. Let's draw that square again:

HCN (set up the screen)

4(25F2R) (make the square)

If you look carefully at the screen, the turtle is now pointed up instead of pointing to the left. Why? (Hint: The first time you made the square, was there a 2R at the end?)

The blue area has another surprise - the entire sequence was entered, and then WSFN did the command. WSFN didn't do each step as you typed them in.

When you enter a parenthesis, WSFN will wait until you enter the other parenthesis before doing the command. When you are doing a long set of commands, a parenthesis will let you see the entire line before WSFN executes it - if you make a mistake, the BACK S key can be used to erase letters. Let's try an example:

(The object is to make a rectangle 20 units high and 10 units wide.)

HCN

(20F2R10F2R20F3R10F oops! you want 2R last time

enter 5 BACK-S , and then enter:

2R10F)

No! Made 1 only...

(As an exercise, take a look at the picture that would result if you didn't correct the error - that's a mighty strange rectangle!)

Nesting Parenthesis

Pairs of parenthesis can be used to build bigger and better commands out of smaller ones. WSFN will wait until all the parenthesis are "balanced" - that is, the number of) matches the number of (. For example, try:

```
((((((((((20F))))))))))
```

The 20F won't be done until the last) is entered.

Here is a nicer kind of square:

```
8(4(20F2R)R)
```

Once this picture is on your screen (You have to actually do this one - we won't put the picture in this manual!), let's see if it can be figured out.

The way to understand lines with parenthesis is to work from the inside out, starting with the innermost pair. In this case, this is:

```
(20F2R)
```

This draws one side of a square, and turns the turtle 90^{o} to the right. Now lets build up the next layer:

```
4(20F2R)R
```

The first part draws the four sides of the square. Then the turtle is turned once to the right. This means that the next time a square is drawn, it will be tilted diagonally.

Now, the entire thing is done eight times:

```
8(4(20F2R)R)
```

If you trace this out by hand, the turtle ends up at the Home spot after each square - but the turtle is turned once to the right each time. When the square was drawn the first time, the home spot is the lower left corner.....

WSFN tends to draw its pictures rather rapidly - to see how a picture is constructed maeans you have to slow WSFN down in some manner. The W command tells WSFN to wait for 1/30 second. Let's add this to the fancy square and see how the square is made:

```
8(4(20F2R30W)R)
```

After each side of a square is drawn, the turtle waits for one second - and it's much easier to see how the picture is made!

Just for fun, try these lines, and see if you can figure out how they work. If they go too fast, insert a 30W inside the innermost parenthesis (like we did with the fancy square).

```
8(8(10FR)R)          (stained glass window)
8(8(5FR)20FR)        (super eight)
8(8(5FR)20FL)        (different for such a small change!)
8(6(5FR)R)
8(7(5FR)R)
```

You can see that small changes can make big differences! Here is a really long line to try:

```
8(8(8(6(5FR)10FL)15FR)20FL)
```

Some Optional Exercises

Try the first few of these to see how complicated shapes can be made from simpler ones. It is quite difficult to type all of these in correctly, so feel free to quit when you get frustrated.

The series of entries shown below shows how to construct a checkerboard by making a square, and then adding more and more until a checkerboard is drawn. In the process, some mistakes (which computer programmers call bugs) will appear - and then they will be fixed. If you try to make complicated pictures of your own, you will make mistakes too.

The new parts of each entry will be underlined.

1) 4(8F2R) This draws a square

2) (4(8F2R)8F) Once a square is made, move the turtle to start a new square.

3) 8(4(8F2R)8F) Do eight squares in a row.

4) (HCN40F4R8(4(8F2R)8F))

The eight squares ran off the top of the screen, so put the turtle near the top and point him downward. Then do the row of squares.

5) (HCN40F4R8(4(8F2R)8F)2R16F2R64F)

After doing a row of squares, put the turtle in position to start a new row.

6) (HCN40F2R32F2R8(4(8F2R)8F)2R16F2R64F)

Just putting the turtle over to the right to center the checkerboard on the screen.

7) ((HCNU40F2R32F2RD)8(4(8F2R)8F)(U2R16F2R64FD))

When the turtle is being put into position, it shouldn't draw a line. The pen is lifted before a move and put down afterward. The new parenthesis separate the positioning moves from the square drawing and help make the entry easier to read.

(Note: If you have got this far, your entry will appear on two lines of the screen's lower area. WSFN can accept up to character lines.)

8) ((HCNU40F2R32F2RD)8(8(4(8F2R)8F)(U2R16F2R64FD))

To make a checkerboard, just repeat the row of squares and the move to the next row eight times - but it doesn't work! (When a row is started, the turtle must point down, and we left it pointing up.)

9) ((HCNU40F2R32F2RD)8(8(4(8F2R)8F)(U2R16F2R64F4RD)))

With the turtle's direction fixed, we get a checkerboard, but with the rows too far apart - a new row starts 8 steps to the left instead of 16. Let's fix this and try again.

10) ((HCNU40F2R32F2RD)8(8(4(8F2R)8F)(U2R8F2R64F4RD)))

Wow! It works this time!!!!

MAKING YOUR OWN COMMANDS

Each time you want to draw a small picture, you have to enter the commands for the turtle - for example, to make three squares means you have to repeat:

```
4(12F2R)          (draw the square)

-------           (move the turtle to the next spot)
```

three times - once for each time the square is drawn. A design with a lot of squares could take a lot of typing!

WSFN lets you make new commands - and it will remember the command as long as the ATARI is turned on (or until you change the command's meaning). Here's how to make a "Square" command:

```
=*S4(12F2R)
```

The = tells WSFN that a new command is being named. The name of the command is *S, and the meaning of the command is 4(12F2R) which draws a square.

Drawing the square is easy now - just enter:

```
*S
```

and the square appears. You can also put your new command into other WSFN lines - for example:

```
8(*SR)

4(8(*SR)24F2R)
```

If you have been doing the examples, you surely have entered this one a lot:

```
HCN
```

Why not make it a command as well - *C, for example:

```
=*C(HCN)
```

You can include commands that you have invented into other commands. Let's take that fancy design and turn it into a command:

```
=*F(*C4(8(*SR)24F2R))
```

Now by pressing just two keys, a complete picture is drawn!

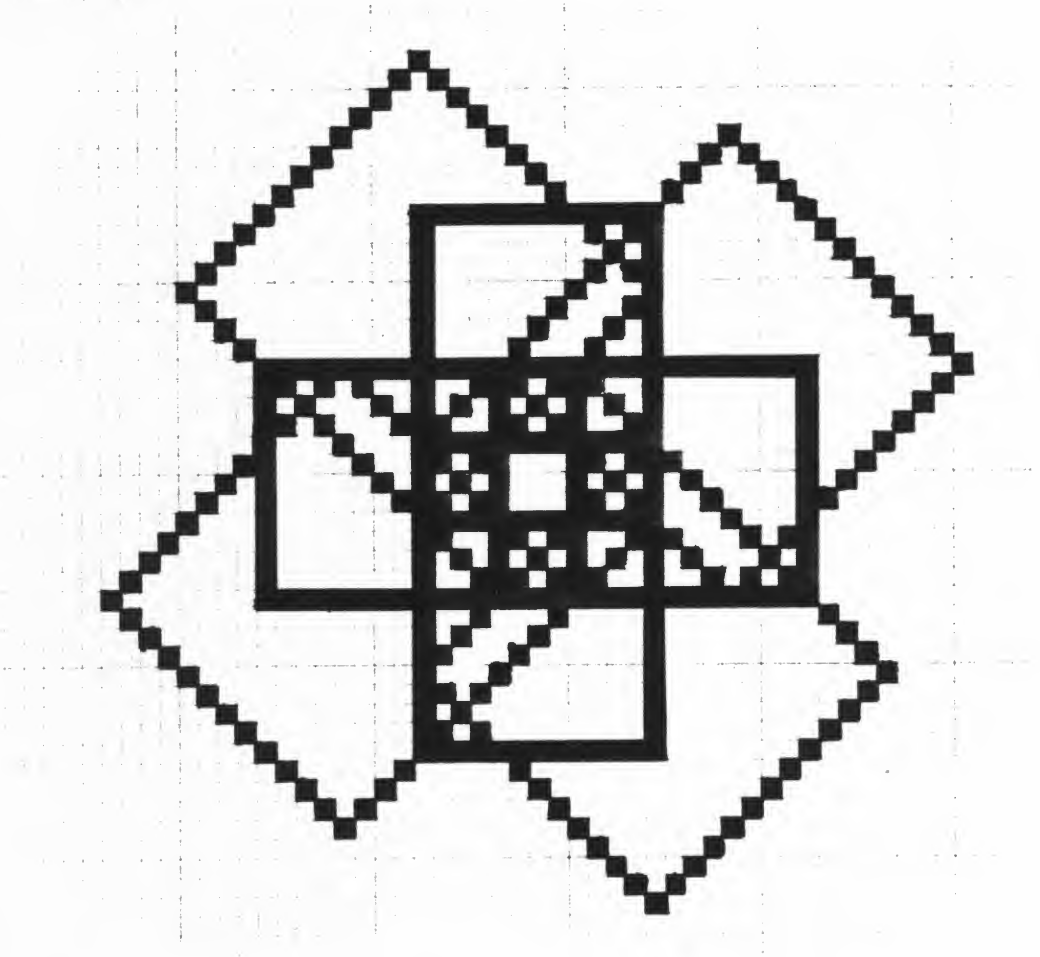
(Notes: 1) WSFN uses the * to tell if a command is already in the "canned" commands that it already knows or if you invented it. 2) An invented command will end in the same way a normal WSFN command does - for example, if you want *Z to mean 4F3R, you should put some parenthesis around it like this: =*Z(4F3R) If you don't, WSFN will think that *Z means 4F. 3) WSFN doesn't actually do the command until you tell it to - the = is a command to record new commands. Use the command (such as *S or *Z) to make WSFN perform.)

A Command That Lasts Forever (Almost)

Let's make *Q a command with its own name included (This should drive WSFN nuts!):

=*Q(*SR4F*Q)

The turtle draws the design below, and then does it over, and over, and over, and .....



The command *Q tells WSFN to draw the square *S, move the turtle and turn it a little , and then, do *Q again! This is called an infinite loop - the same thing gets done forever....

(Note: Actually, WSFN will do *Q for several minutes, and then quit. The reasons for this are beyond the scope of this manual. It is possible to make true infinite loops in WSFN - again, using features that aren't covered here.)

To make WSFN stop, press the BREAK key - BREAK will always stop WSFN no matter what it is doing.

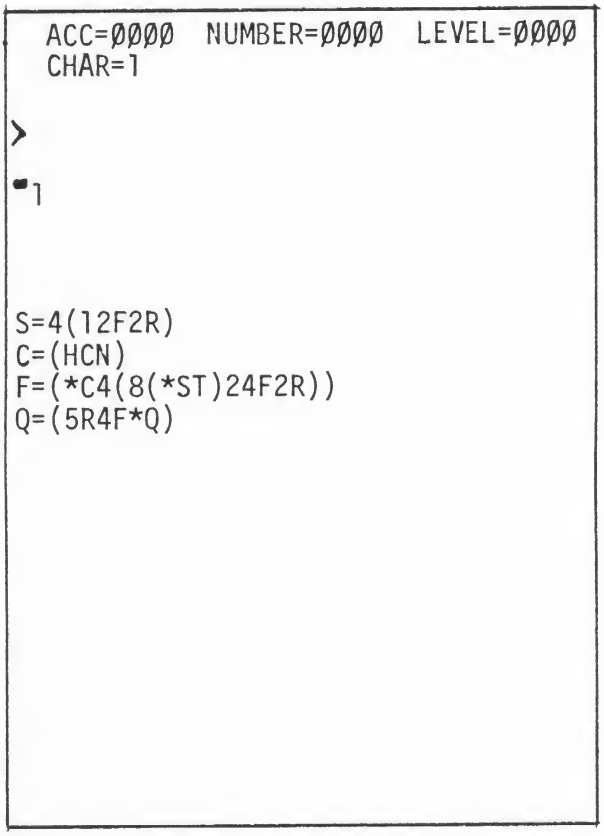
Enter *Q again, and while it is going, press the R or L keys now and then - the figure will move around. WSFN will stop in the middle of a command to do one that you just typed in (in this case, R or L), and then continue doing the old command.

Looking at Your Commands

Now that we have several new commands, how do we look at them if changes are needed. Just enter:

M^C1 (That's CTRL-M and 1)

The screen will change to:



The Debug Mode Screen

This is called the "debugging mode". The four lines in the middle are the four commands you entered.

If you want to change a command, you can either type it in again, or use the cursor movements & screen editor to change the command's line directly. (This manual assumes that you know how to use the screen editor.)

There is one big difference in debugging mode - to make WSFN do a line, you must press the RETURN key.

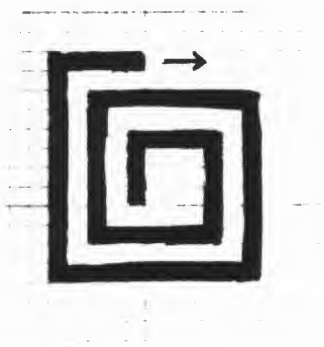
To leave debugging mode, enter:

M^C3 RETURN (CTRL-M, 3, and press RETURN)

(Note: WSFN's debugging mode has many features which won't be discussed in this manual. For the present, just use debugging mode to see your library of special commands, and to make changes to your special commands.)

THE ACCUMULATOR

WSFN has an "accumulator" - which is a counter which can be used to keep track of what you are doing. For example, suppose you want to draw a spiral square like this:



Each side of the square is one step larger than the previous one. If you try to do this with the WSFN you already know, it gets harder and harder to do:

(3F2R)	(one side done)
((3F2R)F(3F2R))	(two sides done)
=*S(3F2R)	(make a side a special command)
(*SF*S2F*S3F*S4F)	(once around)
(*SF*S2F*S3F*S4F*S5F*S6F*S7F*S8F)	
etc	

The accumulator can be used to solve this problem. In WSFN, the commands:

+	adds one to the accumulator.
-	subtracts one from the accumulator.
A	repeats the next command as many times as indicated by the number in the accumulator.

Let's see how this works in practice. Starting with the turtle at home, and pointing north, enter:

++++++++++	(press + ten times)
AF	

The turtle will move up 10 steps - that's the number in the accumulator. If you repeat:

AF

the turtle will move another 10 steps forward.

Try entering several -, and then the AF. Now the turtle will move a shorter distance. If you press - enough times, the accumulator will have a zero in it. Now the turtle won't move at all.

The smallest number in the accumulator is zero (and the largest is 9999). If you enter too many -, the accumulator will just stay at zero. Try pressing - 15 times, and then AF - the turtle stays still.

The commands + and - can be repeated like any others. For example,

15+	adds 15 to the accumulator
23-	subtracts 23 from the accumulator

If you use A to tell how many + or - to use, you will get:

A-	sets the accumulator to zero
A+	doubles the number in the accumulator

(Be sure you understand the A+ and A-. A- is used a lot to "make things ready" - it is easy to enter a command which leaves some junk in the accumulator - and when you try the command again, WSFN will do something different! (and usually not very nice).)

If you want to see what's in the accumulator, go into debug mode and play a bit with the + and - commands. The accumulator is shown in the line that says:

ACC=ØØØØ

in the upper left corner. + and - will change this value, for example,

+++++++ RETURN

will make the accumulator look like this:

ACC=ØØØ7

(Note: M^C1 gets you into debug mode, and M^C3 gets you out.)

Now, back to the square spiral - each side is one unit longer than the previous one, so:

(AF2R+)	will draw a side A units long, and add one to the accumulator.

Here is the spiral:

```
HCNA-40(AF2R+)
```

HCN sets up the screen, A- removes any leftover junk, and 40(AF2R+) draws 40 sides to the spiral.

To fill up the screen, enter:

```
120(AF2R+)
```

Note that the spiral continues from where it stopped before - we didn't zero the accumulator or home the turtle!

Just for fun, try these variations on spirals - and see if you understand what they do. (The really serious student is asked to visualize the picture before trying them out.)

Squares:

```
40(4(AFRR)+)
40((4(AFRR))+)
40((4(AFRR))++)
40((4(AFRR))++RR)
```

It is useful to define this command to set up the screen between pictures:

```
=*R(HCNA-)
```

Octagons:

```
40((8(AFR))+)
40(AFR+)
40(7(AFR)+)
40(6(AFR)+R)
```

Remember that you can insert some delay if you put a 30W inside the innermost parenthesis, for example,

```
40(4(AFRR30W)+)
```

Single Step Mode

Sometimes WSFN will do things much too fast, and you will need to have a closer look at what's happening. (If you are just learning WSFN, skip this part until later.)

There are two commands that can help you look more closely:

M^C2 — This puts the upper part of the debug mode display into the blue area - if you try some of the previous examples, you can see the value of the accumulator change. The line called:

CHAR=_

displays the current character that WSFN is looking at.

If you enter M^C3 you will return to the usual mode. (If you want a screen without any blue area, use M^CØ.)

S^C1 — This is single-step mode. WSFN will do one action each time you press CTRL-4. For example, enter:

10+ (get into debug mode first)

and the accumulator will increment by one each time you press 4^C.

S^C has some other options also:

- S^CØ — returns WSFN to normal speed
- S^C7 — makes WSFN do about 1 action per second.

When you are in either debug mode (M^C2 or M^C1), WSFN will run more slowly - it has the job of writing the debug data on the screen, and this takes some time to do.

Try some of the simpler examples that you already understand with these new commands - and you will see how they can help you understand complicated commands.

TESTING AND DECISIONS

WSFN has several commands which will select between two commands. For example, if the accumulator is zero, one command will be performed. If the accumulator isn't zero, the other command will be done. These commands are known as tests, and they all look like:

```
(test something) (if the result is true, do this one)
                 (if the result is false, do this one instead)
```

The test for the accumulator looks like this:

```
T(10F)(10R)
```

If the accumulator isn't zero, the first action (10F) is chosen. If the accumulator is zero, the second action (10R) is done instead. In both cases, the action that isn't chosen is ignored.

Let's look at some of these in more detail:

The Random Test (?)

The command ? will select one of the two commands following it with 1:1 odds (You can think of WSFN flipping a coin - if the coin is heads, the first command is done, and if it is tails, the second command is done.)

Try the following entry on your ATARI - do it several times. Each time you enter this command, the turtle either moves forward 5 units or turns to the right:

```
?5FR          (Remember that TC2 sets up your turtle.)
```

Repeating this can make some nice squiggles:

```
1000?5FR
```

Note: If you can't get the ? to work, remember that ? is a shifted character - you must press SHIFT to get a ?.

Parenthesis can be used to build more complicated tests:

```
100?(U5?5FR)(D8(5FR))
```

Try it out, and then let's analyze what this expression does. First, there is 100 repetitions of the expression:

??(U5?5FR)(D8(5FR))

The first action is U5?5FR which is selected ½ of the time. This lifts the pen, and moves the turtle randomly five times. (Just the previous examples with the pen up.)

The other action is D8(5FR) which draws an octagon on the screen.

Here are a few other random patterns to play with:

```
100?(10F?RL)(8(3FL)2R)
100?(H20?5F?RL)(2R8(2FR)2L)
100?(H20?5F?RL)(10?(2R8(4FL)2L)12F)
```

The Accumulator Test (T)

The command T checks the accumulator. If the accumulator is not zero, the first action after the T is performed. If the accumulator is zero, the second action takes place.

By setting the accumulator, you can make your commands more responsive - they can check the accumulator and select different actions if the accumulator is zero.

Here is an example:

```
=*ET(20F2R)(4(5F2R))

HCNA-
```

Entering *E will draw a small square (That's the 4(5F2R) when the accumulator is zero.)

If you do a +, *E now moves the turtle 10 units and turns it right - doing this four times makes a larger square.

Counting the nunber in the accumulator can be handy - let's see what can be done with spirals:

```
=*ST(AF2R-*S)(sp)          (sp means SPACE)
```

If something is in the accumulator, draw part of the square spiral. If the accumulator is empty, do the sp - space is WSFN for "do nothing". Here it means we are done with the spiral.

After the side of the square is drawn, subtract one, and then call *S to do the next side - this will continue until the accumulator is zero.

Try setting the accumulator to different values and see the spirals that can be drawn.

If you want to check if the accumulator is larger than 32, use:

32-T(etc)

First, subtract 32, and then test the accumulator.

The Use of Brackets

If you enclose some WSFN commands in brackets [ ], an interesting thing happens - the accumulator is saved, and no matter what you do to the accumulator inside the brackets, it will be restored to its old value when you come outside. The brackets also work like parenthesis for grouping commands together.

Here is an example - we will move the turtle forward by the number in the accumulator, draw the spiral with *S, turn right, subtract 5 from the accumulator, and repeat until the accumulator is empty.

If we used parenthesis around *S, only one spiral would be drawn - why? - the command *S leaves the accumulator at zero. The cure is to put brackets around *S which preserves the accumulator while *S is performed. Let's try it:

(be sure *S is already entered)

=*QT(AF[*S]2R5-*Q)(sp)

To try this out, enter:

HCNA-30+*Q

It's fun to try different values of the accumulator and see what *Q does.

Some Last Optional Exercises

1) Change *S to some of the other spirals and octagons in the section on the accumulator.

2) Change *Q to move about randomly, and then to set the accumulator at random (from 2 to 20) and call *S - and do this 20 times.

3) Look at the optional exercises that draw a checkerboard. Break the expressions down into several * commands which A) draw the square, B) move the square one unit, C) draws a row of squares, D) moves to the next row, and E) does the entire board. Notice how much easier it is to check each part and to fix bugs (mistakes that you made, being a fallible human being).

SOME FINAL COMMENTS

Now that you have survived this introduction to WSFN, there's a lot more to cover. (Get the reference manual for details.) WSFN can:

1) Check the joysticks position and triggers.
2) Check the paddle pots and triggers.
3) Make sounds - both beeps and via the TV set.
4) Change the colors in the display.
5) Check if the square in front of the turtle has been drawn on.
6) Save and load your commands from discs and tapes.
7) Load "canned" commands from the cartridge ROM.
8) Control repetition in several other ways.
9) Represent the turtle - or make it go away in different ways.
10) Control how the turtle treats the edge of the display.
11) Save the accumulator in "variables" for later use.
12) Report some errors in debug mode.

SOME DEMONSTRATION PROGRAMS

The following demonstrations illustrate some of the extra features in WSFN and are fun to play with.

Please take care to see that you have entered them correctly - an error in entry usually means the program won't work. Each group is separate unless stated otherwise.

The Zapper

The Zapper illustrates the use of sound and color options in WSFN. Three procedures, *A, *C, and *D are required.

```
=*A(MCØACGA-1(+P&1*C~))
=*C(&Ø[A+&1][T(AF*D&2-*C&3)H ])
=*D(3R2F2R)
```

Start the Zapper with *A.

Variations of the Zapper involve changes to *D:

```
=*D(?(3R2F2L)(3L2F3L))
=*D[?7(AFR-)(L)]
=*D[T(7(AF2R)FR-)sp]                    sp means SPACE
```

Several of the shapes in the manual, such as the octagonal and square spirals, can be used. Enclose these procedures in square brackets [] to preserve the accumulator.

Some Joystick Exercises

WSFN can look at the joysticks, and these examples show a few possibilities. The joystick is to be plugged into Port Ø (the left-most port in front.)

Each of these joystick programs are started with *J.

Draw Program

This makes the joystick like a turtle. UP moves the turtle forward, RIGHT turns the turtle R, LEFT turns the turtle L, and DOWN puts the turtle at home (H).

```
=*J(MCØTC21(*K^))
=*K($A(FB)sp$B(RB)sp$D(LB)sp$C(H5B)sp)
```

Planting Posies

```
=*J(MCØTC2HN1(*K^))
=*K(U$AFsp$BRsp$DLsp$C(D*PB)sp2W)
=*P8(8(3FR)3L6F)
```

Here, DOWN plants a "posie", and the other controls move the turtle around with the pen up. An alternate "posie" is:

```
=*P(8(8(3FRACAW)3L6F2RACØ))
```

and each "posie" makes a sound!

The Superturtle

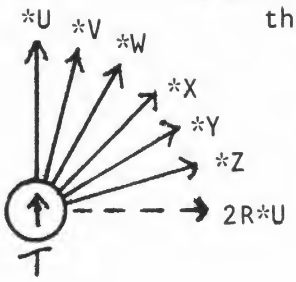
Since the WSFN turtle has only 8 directions, here is a "high resolution" turtle with 24 directions. The commands *F, *R, and *L perform the Superturtle equivalents of F,R, and L. *F will move the turtle about 4 units horizontially/vertically and about 3 units diagonally. (Surely you have noticed that F on the diagonal is longer than F off the diagonal. *F tries to correct this distortion.)

The Superturtle commands will not change the accumulator and can be substituted into other figures. (ie, change F to *F, etc.)

DC7MCØ will set up a high resolution display (2 colors only) in which Superturtle can be used nicely.

```
=*U4F
=*V(2FRFLF)
=*W(FR2FLF)
=*X(R3FL)
=*Y(2RFL2FRF2L)
=*Z(2R2FLFRF2L)
```

*U to *Z draw in each Superturtle direction.



When the turtle is pointed North, the procedures *U to *Z will move as shown. The turtle remains pointed North. To move in the other directions involves using 2R or 2L to get into the next quadrant.

Superturtle Directions

The individual directions should be checked by homing the turtle and entering 8 or 10 repetitions for each line. For example, H8*V to check *V.

```
=*F[A-#Z+T(-T(-T(-T(-T(-Tsp*Z)*Y)*X)*W)*V)*U]
=*R[A-#Z+4-T(A-=#Z2R)(#Z++=#Z)]
=*L[A-#Z+T(-=#Z)(5+=#Z2L)]
```

If the six direction primitives are correct, this expression will draw a nice circle:

```
(*U*V*W*X*Y*Z2R)
```

(Be sure Turtle is not on a diagonal.)

Here are some pictures to do with Superturtle - most are variations of the same idea:

```
24(H*R12*F)
24(H4(*R3*F)3*L)
24(H12(*R*F)11*L)
24(12*F11*R)
(A-30(24(A*F*R)+*R))
24(24(*F*L)*R)
3(24(24(*F*L)*R)3*F)
4(6(H12(*L*F)+P11*R))
```

The Superstick

If you have the Superturtle entered, here is a superjoystick which does the Superturtle's version of the draw program. *J will get things going.

```
=*J(HCNMCØ1($A*Fsp$B*Rsp$D*Lsp$EHsp^2W))
```

The Superstick can be "flown" quite nicely.

Colorpower Machine

Control of the color registers can create some nice effects. The procedures offered here permit the use of two joysticks to control the colors in a drawing.

```
=*A(A-#A++=#A&1)
=*B(A-#B++=#B&2)
=*C(A-#A+-=#A&1)
=*D(A-#B+-=#B&2)
=*E(A-#C++=#C&3)
=*F(A-#D++=#D&Ø)
=*G(A-#C+-=#C&3)
=*H(A-#D+-=#D&Ø)
```

Note: If WSFN had @1 mean load acc with color in color reg 1, these procedures would be much shorter. (ie, =*A(@1+&1) .)

A nice test figure is drawn by invoking *M:

```
=*M(MCØTCØHCNA-=#A=#B=#C=#D*N)
=*N8(3+P[32(8(AFR)+)]R)
```

If you enter the joystick control, invoke *M and then *J, the joysticks will change the colors. Opposite directions (UP - DOWN, RIGHT-LEFT) change a single color.

```
=*J1($A*Asp$B*Bsp$C*Csp$D*Dsp$E*Esp$F*Fsp$G*Gsp$H*Hsp3W^)
```

The color control routines can make some nice effects by being included in other WSFN expressions. Change *N as follows, invoke with *M, and then try: (You might want to use MC3 in *M instead)

```
=*N(150?(+PH)(20(3F?RL)))

1(?(?*A*B)(?*C*D)W^)
(A-1(A[*A]+A[*B]+A[*C]+A[*D]^))
1(????????*A*B*C*D*E*F*G*Hspspspsp^)
1(^?(?(?*A*B)(?*C*D))(?(?*E*F)(?*G*H)))
1(^?(?(16*A)(16*B))(?(16*C)(16*D)6W))
```

NOTES FOR DEMONSTRATIONS

These brief notes will help explain how the various demonstrations operate and the special features used.

The Zapper: *A sets up the screen in full graphic mode (MCØ) and turns on the sound (ACG) and zeroes the accumulator. It then repetitively calls *C. The caret increments the iteration counter. When this is inside a parenthesis preceded by 1 or more, the result is a true infinite loop. The command P sets the turtle's pen color to the value in the accumulator - in WSFN, this is the 4 values selected by the 2 LSB of the accumulator.

*C plays with the color registers &Ø, &1, &2, and &3. The & sets the indicated color register to the accumulator's value. Each color register may have a value from 0 to 255, and the color selected will appear on any lines drawn with the corresponding pen selection (P). *C then moves the turtle A units, invokes *D, and then invokes itself recursively. When the accumulator zeroes, the turtle is homed, and *C is at last finished.

*D is used to draw any convenient figure to embellish the plot.

Joystick Exercises *J sets up the screen with a turtle and repeatedly calls *K with the caret trick.

*K tests each position of the joystick and then the turtle is moved appropriately (ie, FB, RB, etc). The beep is included to provide feedback, and to slow things down to human speed. Home gives a longer beep.

In the posies, *J is a little better with setup of the screen. *K lifts the pen and moves the turtle - except with the DOWN option which puts the pen down and draws the posie. The 2W is needed to slow things down.

The first *P is clear enough. The second one turns the sound on, and then off. (A^C followed with A-P makes some "artifact" sounds with the letter selecting the artifact. A^CØ turns the sound off. Try it with other WSFN drawings.)

$ followed by a letter tests a joystick position and does the first command if the joystick is moved, and the second if it is not. For example, $EFH would move the turtle if the joystick is UP, and home the turtle if the joystick wasn't UP.

The joystick positions are:

UP	RIGHT	DOWN	LEFT	
E	F	G	H	Port Ø
A	B	C	D	Port 1
M	N	O	P	Port 2
I	J	K	L	Port 3

Superturtle: *U to *Z draw one of the 6 directions in each quadrant. Turning the turtle 90 degrees permits re-use of these procedures in the three other quadrants.

*F tests the value of a variable #Z which stores the superturtle's "direction" which is from Ø to 5. It then calls the appropriate drawer from *U-*Z.

*R checks #Z, and does one of:
If #Z is 5, set #Z to zero and turn the turtle
right 90 degrees.
Else add 1 to #Z.

*L checks #Z, and does one of:

IF #Z is sero, set Z to 5 and turn the turtle
left 90 degrees.
Else subtract one from #Z.

WSFN permits the storage of the accumulator's value into variables, identified by # (#A to #Z). The two operations on variables are:

=#var Set var to value in accumulator
A-#var+ Set accumulator to variable's value

Variables can also be used for repetitions (as shown in setting the accumulator above.)

Superstick: Done by substituting the Superturtle *F, *R and *L into the draw program and putting the loop & setup into *J also.

Colorpower Machine: Procedures *A through *H increment or decrement the color registers. Variables #A to #D hold the color register values.

*M is used to set up the screen - and! to initially define the variables used. *N draws an abstract pattern which has many textures.

*J is an extended version of the usual joystick text - now handling two joysticks in Port Ø and Port 1. When running, three of the colors will be changed. One set is the background, and cannot change (at present). As the values in #A to #D increase, the program runs more slowly due to the #A+ (etc) needed to set up the accumulator. The colors in the color registers are arranged in a way that changes brightness faster than hue which takes getting used to.

The various expressions randomly change the color values by different ruses and will run indefinitely. *N was changed for the heck of it.

WSFN/TURTLE BUILT IN PROGRAMS

The Atari WSFN/TURTLE cartridge has several built in programs (user commands) which are accessible using the CTRL-L or the CTRL-R commands. These programs are itemized in Appendix G of the language/system specification. The paragraphs that follow will elaborate on the information contained in Appendix G.

1. The column labeled "Load/Run Name" contains the one character name of the program group to be loaded by CTRL-L or CTRL-R. See page 15 of the specification for the semantics of CTRL-L and CTRL-R.

2. The column labeled "Content" specifies the command name(s) and description(s) of the top level command(s) contained in each group. Note that, in general, each group contains many low level user command definitions and one or two top level commands.

3. The column labeled "Runs" indicates which of the loaded commands will be executed when the CTRL-R command is invoked. For example, CTRL-R 1 is the same as CTRL-L 1 Y.

4. The column labeled "Reference" contains a reference to supporting narrative information, when available.

The remaining paragraphs will give additional information regarding the built in programs themselves. Note also that the content of the groups may be examined by loading them while in the debug mode (CTRL-M 2) or by putting the loaded definitions to the printer (CTRL-P "P:").

SIERPINSKI CURVE -- no additional information required.

HILBERT CURVE -- no additional information required.

TRINARY TREE -- no additional information required.

SIMPLE SPIRAL -- no additional information required.

SUPER SPIRAL -- first draw a complex, non-closed figure on the screen using DRAW (CARTESIAN), and then invoke SUPER SPIRAL.

DRAW (CARTESIAN) -- joystick Ø is used to produce line drawings; the stick controls the cursor motion and the trigger controls pen up/down.

DRAW (POLAR) -- similar to DRAW (CARTESIAN) but uses spaceship type control; it is very difficult to use.

WALL BANGER -- first draw a complex, closed, maze-like figure on the screen using DRAW (CARTESIAN), and then invoke WALL BANGER.

HOLLYWOOD SQUARES -- no additional information required.

KOCH CURVE -- no additional information required.

THE ZAPPER through COLORPOWER -- see "ATARI WSFN, an introduction....".

Note that the audio select command (CTRL-A) may be invoked while another command is executing without affecting the operation of that command; the same holds true for the turtle representation selection (CTRL-T).

Note also that additional demonstration programs can be stored and loaded from a mass storage peripheral such as cassette or disk using the get (CTRL-G) and put (CTRL-P) commands.