

TABLE OF CONTENTS

1. Turtle overview

- 1.1 Turtle's domain
- 1.2 Drawing
- 1.3 Control of turtle
- 1.4 Keyboard control
- 1.5 User defined commands

2. Command forms

- 2.1 Single letter
- 2.2 Testing commands
- 2.3 Iteration
- 2.4 Nesting
- 2.5 Command/value
- 2.6 Definition

3. Specific commands

- 3.1 Turtle position & orientation
- 3.2 Turtle "senses"
- 3.3 Turtle manifestations
- 3.4 Turtle world
- 3.5 Turtle arithmetic
- 3.6 Iteration
- 3.7 Nesting commands & clauses
- 3.8 User defined commands & variables
- 3.9 Random test
- 3.10 Mode control & options
- 3.11 No-ops

4. Examples

- 4.1 Set ACC = \emptyset
Set ACC = 23
- 4.2 Set ACC = ACC *2
- 4.3 Set ACC = ACC *7
- 4.4 Set ACC = ACC / 7
- 4.5 Set pen color to 1 without altering ACC
- 4.6 Find upper right corner of screen
- 4.7 Recurse by number in ACC
- 4.8 Sierpinski Curve
- 4.9 Hilbert Curve
- 4.10 Trinary tree structure

TABLE OF CONTENTS (cont.)

4. Examples (continued)

- 4.11 Spirals
- 4.12 Super spirals
- 4.13 Diagonal plaid
- 4.14 Random pattern #1
- 4.15 Random pattern #2
- 4.16 Random pattern #3
- 4.17 Koch Curve
- 4.18 Hilbert Curve (written in Pascal)

APPENDICES

- Appendix A - List of commands
- Appendix B - Error status codes
- Appendix D - Pen selects (by screen mode)
- Appendix E - Color register values
- Appendix F - Joystick trigger tests
- Appendix G - ROM resident user commands
- Appendix H - Audio select options

1. TURTLE OVERVIEW

The turtle package provides the user with the ability to generate screen graphics using a few simple keyboard commands. The drawing element is called a turtle, which may move around the screen leaving "tracks" (or not, at the user's discretion). His motion and direction are controlled with five commands, each of which is a single keystroke:

H - Place turtle on home position (center of screen).

N - Point turtle north (up).

R - Rotate turtle 45° clockwise.

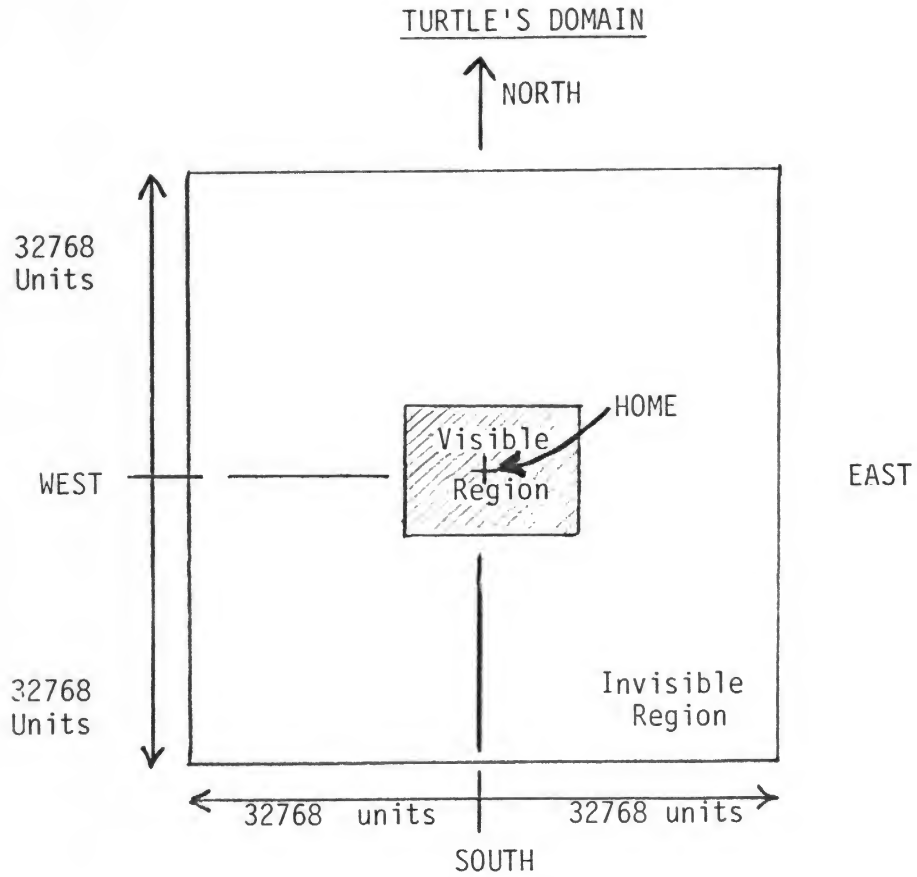
L - Rotate turtle 45° counterclockwise.

F - Move turtle forward one unit.

These commands are sufficient to draw pictures of any degree of complexity, when coupled with other commands and capabilities. Beyond the command set provided by the turtle package, the user may define additional commands which are conglomerations of other system or user commands. Nested and/or recursive command structures may be defined which behave very much like programs in more procedure-oriented languages.

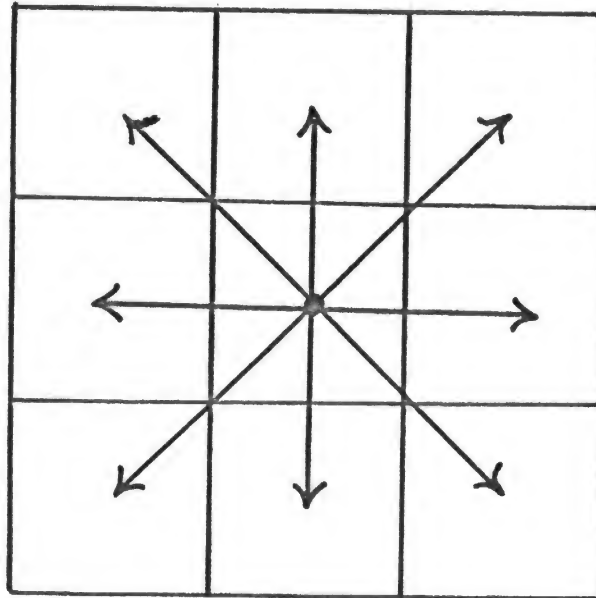
1.1 TURTLE'S DOMAIN

The turtle's domain consists of two regions: (1) the visible region and (2) the invisible region. The domain is fixed in size as a 65536 by 65536 "sphere" with the visible region being variable in size, depending on the current display mode and resolution. The turtle may be confined to the visible region or not, at the user's discretion.



Note that the domain is "spherical" in that the East & West edges are joined as are the North & South edges.

The turtle is always defined by his position and orientation. From his position, he may move to any of 8 adjacent "cells" based upon his orientation, as shown below.



In addition to moving to an adjacent cell, the turtle may also "sense" whether a cell is in the visible region or not, and detect the presence of a "track" in an adjacent cell.

1.2 DRAWING

The turtle draws by leaving "tracks" with a magic marker which is controlled with three commands.

U--which stands for pen up; this allows the turtle to move without leaving tracks.

P--is the pen select command; this selects one of three colors for the pen, or erase (which removes tracks that the turtle crosses).

D--which stands for pen down; this causes the turtle to leave tracks of the current pen color.

1.4 KEYBOARD CONTROL

The keyboard is monitored at all times, even while a command is being executed; if a key is pressed, the current command is temporarily suspended, the newly entered command is executed, and then the prior command is resumed. This command interruption may actually be accomplished to any number of levels (commands interrupting commands interrupting commands, etc.).

The BREAK key will stop all command activity and put the turtle into an idle state.

1.5 USER DEFINED COMMANDS

The user may define unused keys to be "macro" commands--conglomerates of intrinsic and/or user defined commands. These commands may recurse, if desired, and may also, in turn, define other commands or redefine themselves.

2. COMMAND FORMS

Turtle commands fall into one of six forms, as will be described in the paragraphs that follow.

2.1 SINGLE LETTER

`<command> ::= <name>`

Many commands are of this form, where a single key stroke (the name) is the command. Examples are: H (Home), C (Clear), F (Forward).

2.2 TESTING COMMANDS

`<command> ::= <name> <then> <else>`

There are four commands of this form, where a test is made and either the `<then>` clause or the `<else>` clause is executed, (but not both). Examples are: T (Test Acc.), E (Edge test) and ? (Random test).

2.3 ITERATION

`<command> ::= <value> <clause>`

There are three commands of this form, where a numeric value is used to iterate a following clause. Examples are: A (Iterate by ACC), # v (Iterate by variable), and n (Iterate by number).

2.4 NESTING

$$\langle \text{command} \rangle ::= \langle \text{nesting bracket} \rangle \langle \text{clause} \rangle . . \langle \text{clause} \rangle \langle \text{matching bracket} \rangle$$

There are two nesting bracket sets: (...) and [...].

2.5 COMMAND/VALUE

$$\langle \text{command} \rangle ::= \langle \text{name} \rangle \langle \text{value} \rangle$$

2.6 DEFINITION

$$\begin{aligned} \langle \text{command} \rangle & ::= \langle \text{name} \rangle [*] \langle \text{clause} \rangle \\ & ::= \langle \text{name} \rangle \# \end{aligned}$$

3. SPECIFIC COMMANDS

3.1 TURTLE POSITION & ORIENTATION

- Home Turtle - H

Puts turtle to the center of the screen and leaves his "track" if the pen is down; does not alter turtle orientation.

- Point Turtle North - N

Faces the turtle towards the top of the screen without altering his position.

- Turtle Forward - F

Moves the turtle forward one unit and leaves his "track" if the pen is down; does not alter turtle orientation unless the screen edge is hit in reflect mode.

- Rotate Turtle Right - R

Rotates the turtle clockwise 45° .

- Rotate Turtle Left - L

Rotates the turtle counterclockwise 45° .

3.2 TURTLE "SENSES"

- Sense Color Value - S

Reads the color of the square in front of the turtle and puts it into the turtle accumulator (ACC). Background is read as zero, turtle tracks are read just as they were written, and squares beyond the screen edge are read as zero.

3.2 TURTLE "SENSES" (continued)

- Joystick/Trigger Sense - $\$ \langle \text{select} \rangle \langle \text{on} \rangle \langle \text{off} \rangle$

Tests the selected joystick position, joystick trigger and executes either the $\langle \text{on} \rangle$ clause or $\langle \text{off} \rangle$ clause, based upon the result of the test. See Appendix F for a list of the select codes for the various sensible items.

Joystick position and triggers are continuously sensed; there is no edge detection or reset logic.

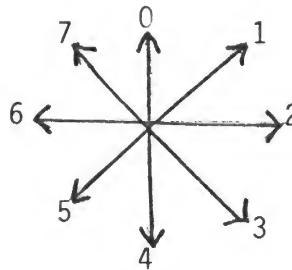
- Pot Controller Read - $\% \langle \text{select} \rangle$

Reads the selected pot controller and puts its value in the accumulator; the controller range is from 0 (full counter-clockwise) to 228 (full clockwise). The $\langle \text{select} \rangle$ values range from 0 to 7 corresponding to the 8 pot controllers.

- Orientation Sense - $;$

Sets the accumulator to one of the values shown below, based on the current orientation:

- 0 - N
- 1 - NE
- 2 - E
- 3 - SE
- 4 - S
- 5 - SW
- 6 - W
- 7 - NW



3.2 TURTLE "SENSES" (continued)

- Edge Test - E <true> <false>

Tests to see if the square in front of the turtle is at or beyond the screen edge; if so, the <true> clause is executed; otherwise, the <false> clause is executed.

3.3 TURTLE MANIFESTATIONS

- Beep - B

Generates an audible tone using the television sound system.

- Pen Up - U

Allows the turtle to move without leaving tracks; this command is countermanded by the "D" command described below.

- Pen Down - D

Lowers the pen, which means that the turtle will leave tracks (or erase), depending upon the current pen color selected.

- Audio Control - CTRL-A <letter>

Allows sounds to be generated as manifestations of internal registers of the turtle program, as shown in Appendix H.

- Wait - W

Causes the turtle to wait for the next 30 HZ clock tick before resuming execution of turtle commands. See also 3.10 for a description of the speed control command.

3.3 TURTLE MANIFESTATIONS (continued)

- Pen Select - P

The value of the turtle accumulator modulo 128 is used as the color select for turtle tracks, until changed. Zero is background (erase) and is not the same as issuing a "PEN UP" command.

- Turtle Representation - CTRL-T <number>

Selects or de-selects a turtle overlay that nondestructively shows turtle position and orientation at all times.

<number> = \emptyset , de-selects the overlay;

<number> = 1, selects an arrow overlay;

<number> = 2, selects a turtle overlay;

<number> = 3, selects a point overlay.

3.4 TURTLE WORLD

- Clear Screen - C

Clears all turtle tracks from the screen without altering the turtle position or orientation.

- Edge Rule Select - CTRL-E <number>

Selects one of four rules to follow when the turtle encounters the edge of the screen.

number = \emptyset , turtle stops;

number = 1, turtle wraps to opposing edge;

number = 2, turtle bounces (reflects) off edge;

number = 3, turtle goes off edge but does not leave tracks until he gets back in screen boundary.

3.4 TURTLE WORLD (continued)

If the turtle is not in the screen boundaries when an edge rule is selected, he will be put to the home position.

- Display Mode Select -

Preselects one of the eight display modes available (see Appendix C for descriptions of each mode). The mode change takes place with the next operating mode select (CTRL-M, see 3.10).

- Color Register Values -

Allows the hardware color register selected to be updated with the value in the turtle accumulator. See Appendix D for the utilization of the color registers for each screen mode and see Appendix E for color register values to use to get the desired colors and luminescence.

3.5 TURTLE ARITHMETIC

- Increment Turtle Accumulator -

Adds one to the four-digit ACC; will not increment it past all nines.

- Decrement Turtle Accumulator -

Subtracts one from the four-digit ACC; will not decrement it past zero.

- Test ACC for ≠ Zero -

Tests to see if the ACC is greater than zero; if so, the clause is executed; otherwise, the clause is executed.

- Set ACC to Zero -

Sets the ACC to zero.

3.6 ITERATION

- Iterate by Constant - <constant> <clause>

Iterates the <clause> by the number indicated in the constant. If the constant is zero, the <clause> is not executed at all. If <constant> exceeds four digits in length, the last four digits are used.

- Iterate by Variable - # <variable> <clause>

Same as above, except current value of the indicated variable is used instead of a constant.

- Iterate by Turtle Accumulator - A <clause>

Same as for CONSTANT, except current value of the ACC is used instead of a constant. The <clause> may modify the ACC without changing the iteration count.

- Stop Iteration - !

When executed, makes the current iteration the last iteration (set the iteration count to zero) within the current iteration level. This command will not affect outer level iteration counts.

- Increment Iteration Count - ^

When executed, increments the current iteration count. Has no effect if not executed within an iteration.

3.7 NESTING COMMANDS & CLAUSES

- Basic Nest - (<clause> ... <clause>)

Any number (including zero) of <clauses> may be nested together creating a single new clause, using matched parentheses. Prens may exist within other prens to any level desired.

3.7 NESTING COMMANDS & CLAUSES (Continued)

- Accumulator Save/Restore Nest - [...]

Behaves the same as above except that the turtle accumulator is saved and restored by the [and] commands.

3.8 USER DEFINED COMMANDS & VARIABLES

- Define User Command - = <name> <clause>
= *<name><clause>

Creates a definition in memory whereby the <clause> may be invoked merely by using the <name>; the optional * allows <name> to be the same as one of the turtle intrinsic commands.

If the <clause> consists of a blank character (NOP), the <name> will be removed from the user command directory.

- Invoke User Command - <name>
* <name>

Once defined, a user command may be used exactly as an intrinsic command is used; however, if <name> is the same as an intrinsic command, the * must be used, as intrinsic commands have a higher priority in the name search.

- Get User Command Definitions - CTRL-G "<name>"

This command clears the current set of user commands and reads in a new set from the indicated device. The device name is in the standard format, e.g., "C:", "D:HILBERT", etc.

- Put User Command Definitions - CTRL-P "<name>"

This command writes the current set of user commands to the indicated device. The device name is in the standard format, e.g., "C:", "P:", "D:HILBERT", etc.

3.8 USER DEFINED COMMANDS & VARIABLES

- Load ROM Resident Command Definitions - CTRL-L <character>

This command clears the current set of user commands and reads a new set from ROM. The character following the command indicates which of several sets to load. See Appendix G for a list of command sets and their content.

- Run ROM Resident Command - CTRL-R <character>

This command performs all of the functions of the LOAD command described above, and in addition, executes one of the commands loaded. See Appendix G for more information.

- Clear User Variables - CTRL-C

Removes all user variables from memory.

- Define User Variable - = # <name>

Creates a definition in memory whereby the current value of the turtle accumulator is assigned to the indicated <name>. Any character may be used as a <name>. See section 3.6 for the use of variables.

3.9 RANDOM TEST

- Random Test - ? <then><else>

Randomly executes either the <then> or the <else> clause, using a hardware random bit generator.

3.10 MODE CONTROL & OPTIONS

- Edge Rule - CTRL-E <number>

- CTRL-E 0 = Stop at edge;
- CTRL-E 1 = Wrap at edge;
- CTRL-E 2 = Reflect at edge;
- CTRL-E 3 = Disappear at edge.*

- Speed Control - CTRL-S <number>

- CTRL-S 0 = Run full speed*;
- CTRL-S 1 = Single step (press CTRL-4 to step);
- CTRL-S 2-7 = Run with delays:
 - 2 = 30 commands/sec.
 - 3 = 15 commands/sec.
 - ⋮
 - 7 ≈ 1 command/sec.

* power-up default

3.10 MODE CONTROL & OPTIONS (continued)

- Operating Mode - CTRL-M <number>

- CTRL-M Ø = Draw mode (full screen graphics) ;
- CTRL-M 1 = Debug mode (full screen text);
- CTRL-M 2 = Split screen with internal registers;
- CTRL-M 3 = Normal mode (Split screen with input echo only).^{*}

DRAW MODE: In draw mode, the entire screen is dedicated to turtle graphics. Commands are executed as they are entered and the user is typing "blind" (the commands are executed, but there is no echoing of the command name to the screen).

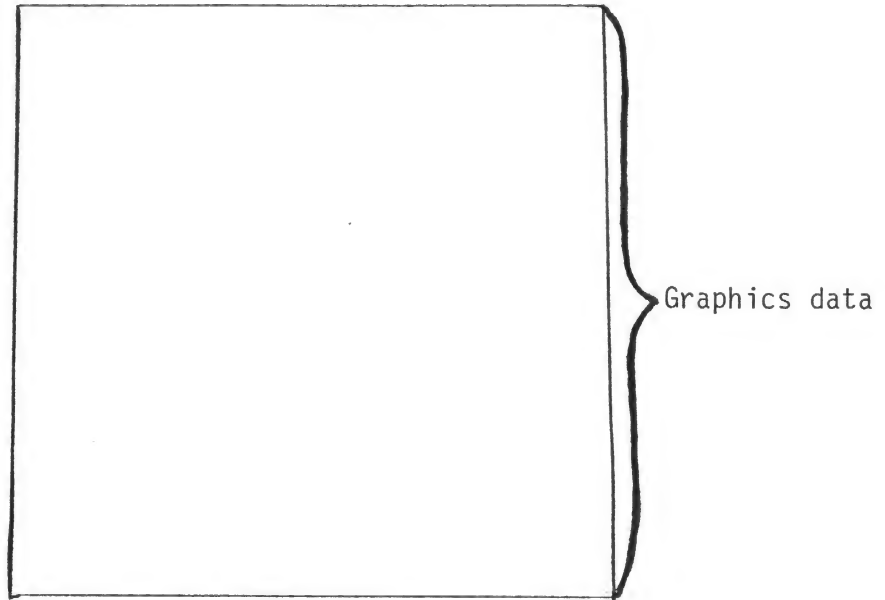
DEBUG MODE: In debug mode, the entire screen is dedicated to text data. Commands to be executed must be entered using the RETURN KEY and line editing may be performed upon input command lines. The screen shows the internal workings of the turtle and also shows all user defined variables and command definitions.

SPLIT-DEBUG MODE: In split-debug mode, the upper portion of the screen is dedicated to turtle graphics and the lower portion contains four text lines. The first two text lines show the turtle registers as in debug mode and the other two text lines are for command entry.

* power-up default

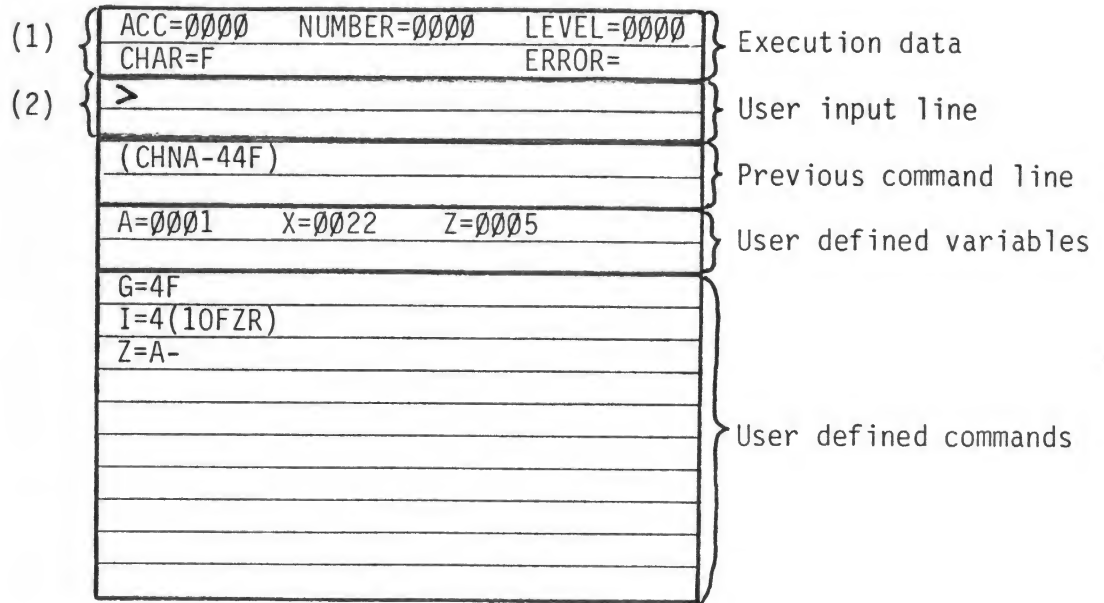
NORMAL MODE: Normal mode is similar to split-debug except that the text portion of the screen is used solely for command entry. When one complete command has been entered, the cursor will move to the beginning of the next line (this may cause the text to scroll); when the command has finished the execution, the cursor will move to the beginning of the next line.

DRAW MODE SCREEN (MODE 0):



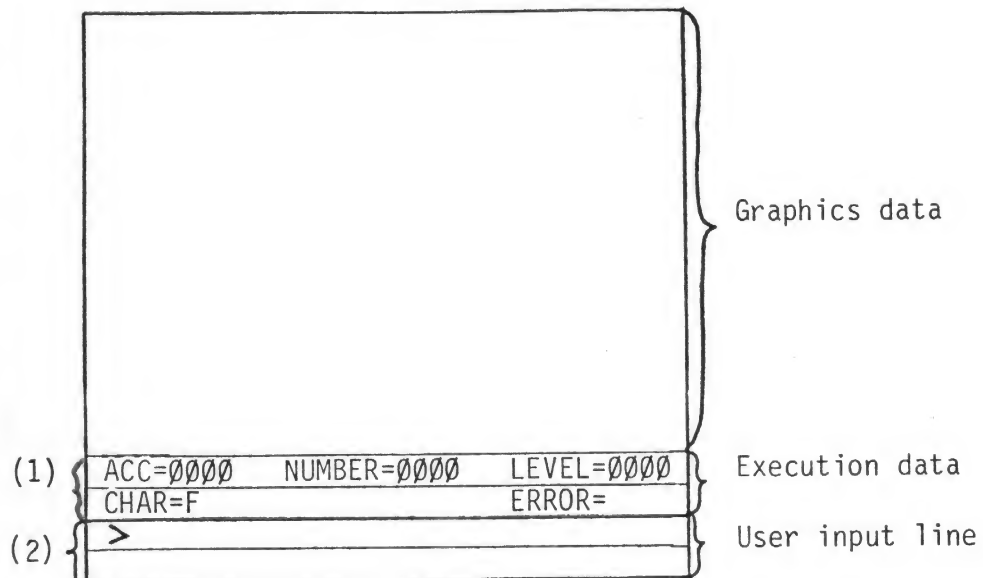
Turtle accepts inputs as typed without echoing them to the screen; no line editing is allowed.

DEBUG MODE SCREEN (MODE 1)

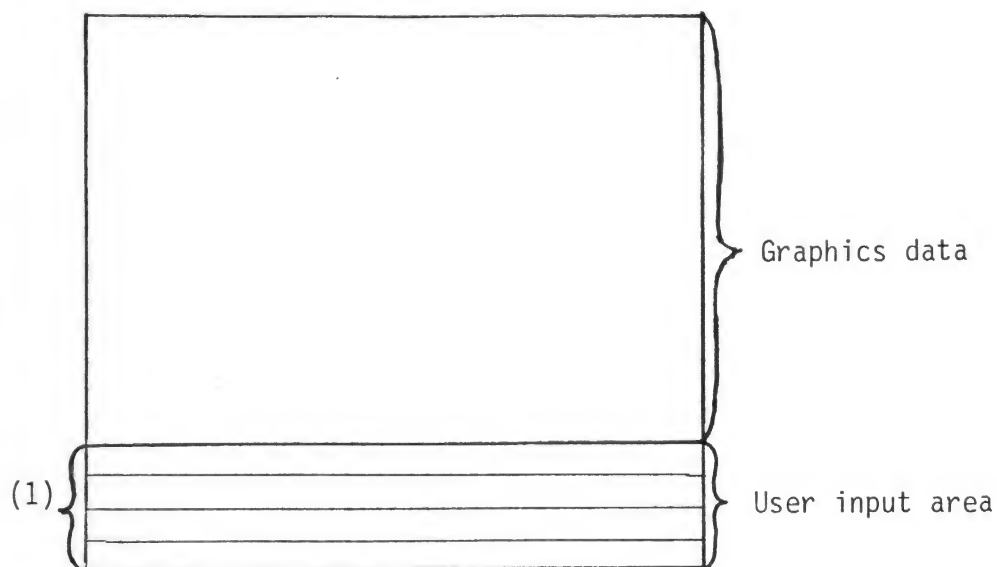


- (1) ACC = Turtle accumulator value
- NUMBER = Iteration counter value
- LEVEL = User defined command "call" level
- CHAR = Current (or last) executed command
- ERROR = Command error code (see Appendix B)

(2) Turtle accepts inputs when ">" is present; ">" disappears while a command is executing. Input lines must be terminated by the RETURN key; line editing is allowed.

SPLIT-DEBUG MODE SCREEN (MODE 2)

- (1) See prior page for explanation.
- (2) Turtle accepts inputs when ">" is present; ">" disappears while a command is executing. Commands are executed as input and no line editing is allowed.

NORMAL MODE SCREEN (MODE 3)

- (1) Commands are scanned as input and no line editing is allowed. The cursor moves to the beginning of the line below the input line (or the input line scrolls upward) when a command is accepted and does the same again when command execution is complete.

- Display Mode - CTRL-D <number>

CTRL-D 0 - 7 = Map to O.S. display modes
1-8, where higher numbers
represent increasingly
higher resolution display.

The system defaults to CTRL-D 6 (Display Mode 7).

This command acts as a pre-select; the mode change actually occurs at the next CTRL-M command.

3.11 NO-OPS

Blank, underscore, and all unassigned keys are treated as no-operation codes.

4. EXAMPLES4.1 Set Acc = \emptyset

A-

4.1 Set Acc = 23

A-23+

4.2 Set Acc = Acc * 2

A+

4.3 Set Acc = Acc * 7

A(6+)

4.4 Set Acc = Acc / 7

=ZT(7-Z+) (+Z-)

4.5 Set pen color to 1 without altering Acc

[A-+P]

4.6 Find upper right corner of screen

=XE (FX) (UNX2RXD)

4.7 Recurse by number in Acc

=ZT(-Z+) Z

4.8 Sierpinski Curve

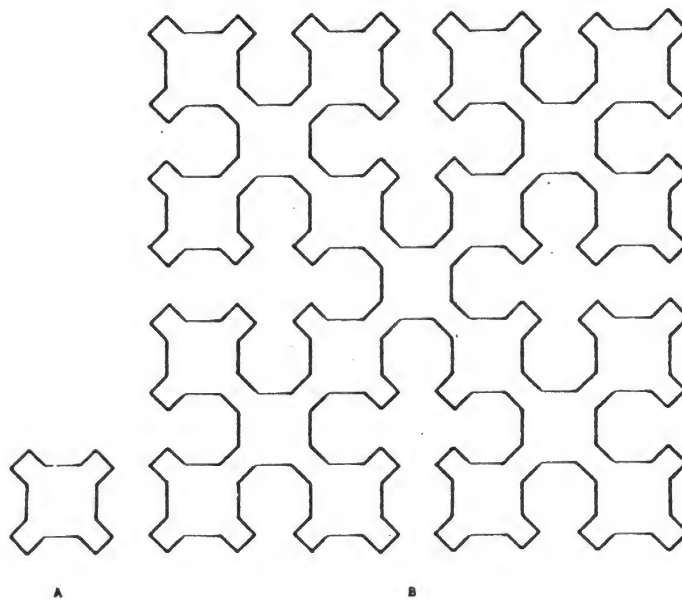
```

=IT(-I2FI3LG3LI2FI+)2R
=G4F
=Y(HNU44F2R44FRC[A-+P]4(2FI))

```

After these are defined, one sets the accumulator to the desired order and types "Y" to draw the curve. Again, "G" can be redefined to a smaller number of "F"s to draw the higher order "Y"s. For "DGOF," the accumulator can be set to 5.

The Sierpinski curve is closed and consists of four identical sides arranged in different directions and connected by a short line "2F." The sides are defined (recursively again) in the macro "I" and the closed curve itself is defined in the last part of the macro "Y," namely: "4(2FI)."

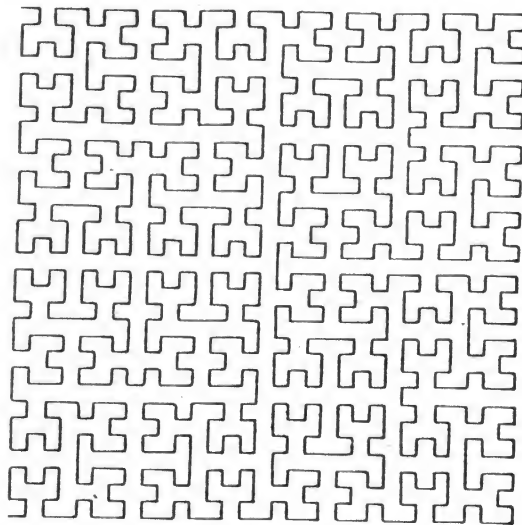


Sierpinski curve of (A) order 1, and (B) order 3.

4.9 Hilbert Curve

```
=ZT(-VG2LZ2RGZG2LV+)2L
=VT(-Z2RGVG2LVZRGZ+)2R
=G4F
=J(HNU44F2R44FC2RDZ)
```

After these are defined, one sets the accumulator to the desired order and types "J" to draw the curve. "G" can be redefined to a smaller number of "F"s to draw higher order curves.



Hilbert curve of (A) order 1, and (B) order 5.

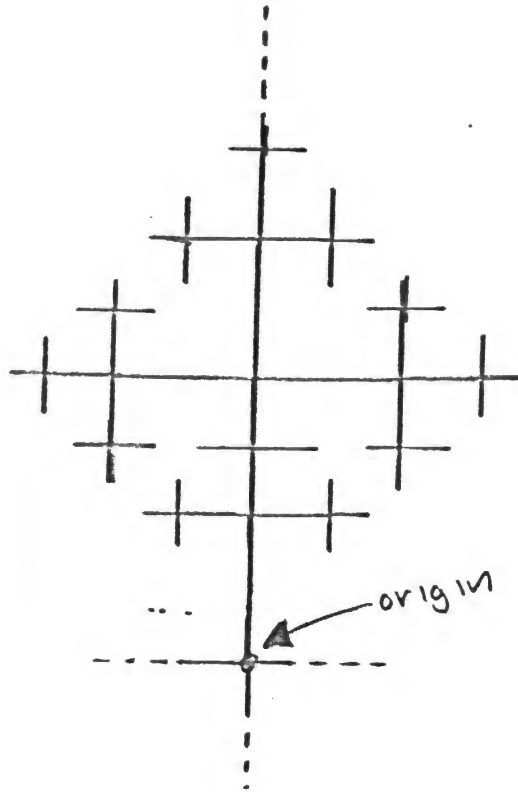
4.10 Trinary Tree Structure

```

=G(-T(++Z-AF2R3(G2R)AFA+)(+4R))
=ZT(2-Z+)
=J(CHNA-4799(4(G2R)A+))

```

Type "J" to Start drawing; press the BREAK key to stop.



4.11 Spirals

```

=I(T(--2(2LAF)Z)_ )
=V(++2(2RAF))
=J[#IY4RU2FD4RZ]

```

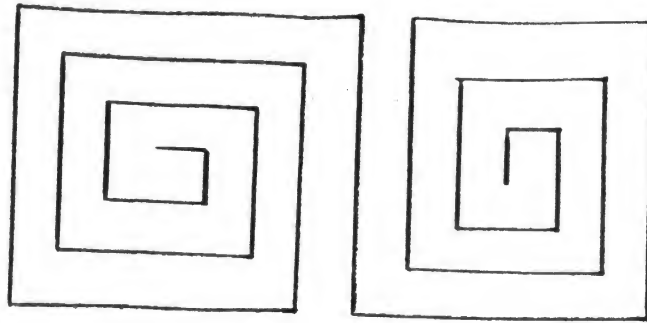
To run: Set accumulator = 0. (Type "A-").

Set #I to spiral iteration count.

(Type "[n+#I]" -- try n=13 first)

Type "Q".

4.11 Spirals (continued)



#I = 7

4.12 Super Spiral

```
=Q (2RIT(LIT(LIT(LIT(LITZ_))_))_)
=I (UFA-2L5TR(SR)RTFD4R)
=Z (B2LY)--initiate backtrack
=Y (2RJT (LJT(LJT(LJT(LJT5B_))_))_)
=J (ST (FY)I)
```

To run: Put obstacles on screen.

Type (HN9999Q)

The command will produce a clockwise spiral figure which avoids all obstacles in its path. It is basically an edge follower with a backtrack algorithm which is invoked when there are no forward moves possible.

4.13 Diagonal Plaid

```
[NR999(10F+P)]
```

4.14 Random Pattern #1

```
(100(8?R2F10F))
```

4.15 Random Pattern #2

```
999?RF
```


4.16 Random Pattern #3

```
100(8?2R_10?F_)
```

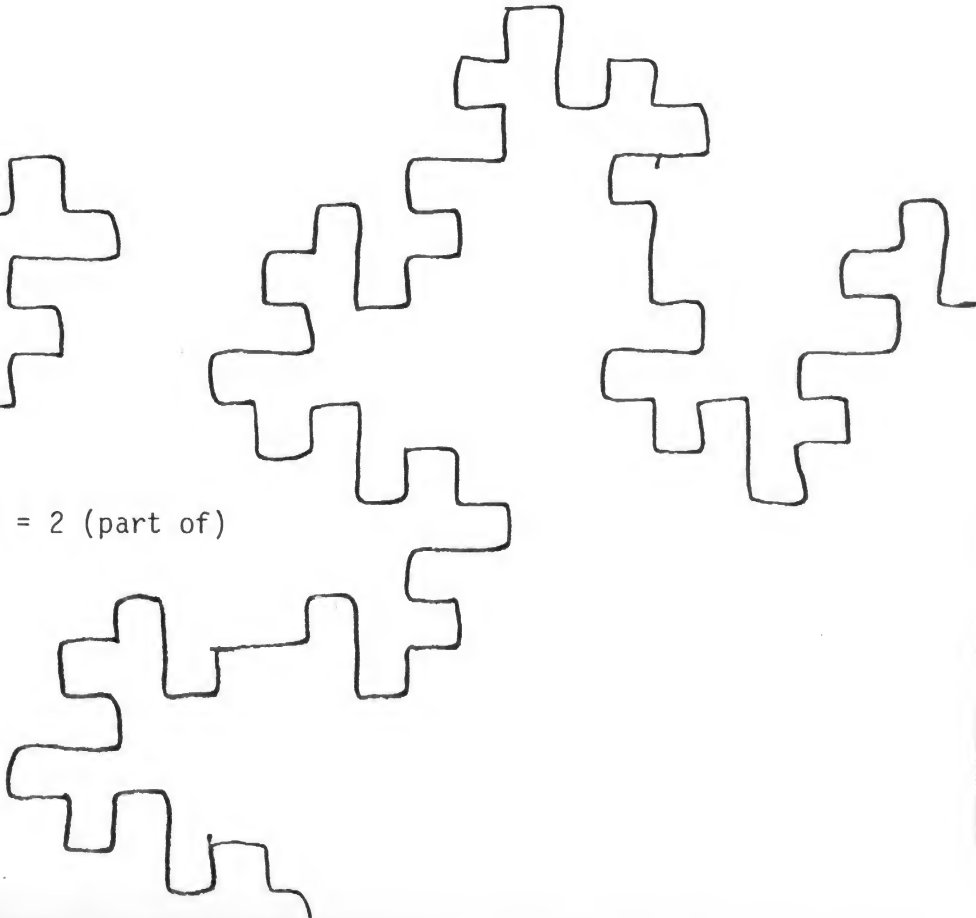
4.17 A Koch Curve

```
=ZT(-ZG4L3(2RGZG)3(GZG2L)4RGZ+)_  
=G2F  
=J4(GZG2R)
```

After those are defined, one sets the accumulator to the desired order and types "J" to draw the curve. "G" can be redefined to a smaller number of "F"s to draw higher order curves.

A = 0 

A = 1 

A = 2 (part of) 

Hilbert Curve (written in Pascal)

```

PROGRAM HILBERT;
VAR SIZE, DELTA, N: INTEGER;
    ORDER: INTEGER;
PROCEDURE HIL(I: INTEGER);
VAR A, B: INTEGER;
    PROCEDURE HIL1;
    BEGIN
        TURN(A); HIL(-B); TURN(A);
    END (*HIL1*);
    PROCEDURE HIL2;
    BEGIN
        MOVE(SIZE);
        HIL(B);
        TURN(-A); MOVE(SIZE); TURN(-A);
        HIL(B);
        MOVE(SIZE);
    END (*HIL2*);
BEGIN (*HIL*)
    IF I=0 THEN TURN(180)
    ELSE
        BEGIN
            IF I>0 THEN
                BEGIN
                    A:=90; B:=I-1;
                END
            ELSE
                BEGIN
                    A:=-90; B:=I+1;
                END;
            HIL1; HIL2; HIL1;
        END;
    END (*HIL*);
BEGIN (*MAIN PROGRAM*);
WRITE('SIZE:');
READLN(SIZE); (*ENTER SIZE FOR YOUR SCREEN*)
WRITE('ORDER:'); READLN(ORDER);
PENCOLOR(NONE);
N:=ORDER-1;
DELTA:=SIZE;
WHILE N>0 DO
    BEGIN (*COMPUTE STARTING (X,Y) POSITION *)
        DELTA:=DELTA*2;
        N:=N-1;
    END;
MOVETO(-DELTA, -DELTA);
PENCOLOR(WHITE);
HIL(ORDER);
END.

```

APPENDIX A - LIST OF COMMANDS

Command Character	Command Syntax	Command Semantics	Paragraph
A	A<command>	Iterate command by value of ACC	
B	B	Beep	
C	C	Clear screen	
D	D	Pen down	
E	E<then><else>	Tests for turtle at screen edge	
F	F	Turtle forward one unit	
H	H	Turtle to home position	
L	L	Rotate turtle left 45 ⁰	
N	N	Point turtle north	
P	P	Value in ACC is pen color select	
R	R	Rotate turtle right 45 ⁰	
S	S	Color select in front of turtle to ACC	
T	T<then><else>	Test ACC for non-zero	
U	U	Pen up	
W	W	Wait for next 30 HZ CLOCK TICK	
<blank>	<blank>	No-op	
!	!	Stop innermost iteration	
#	#<variable><command>	Iterate command by value of variable	
\$	\$<select><then><else>	Test selected joystick position	
%	%<select>	Read selected pot controller to ACC	
&	&<select>	Value in ACC goes to selected color register	
((<command>....)	Binds command as a unit	
@	@	Set ACC to zero	
;	;	Sense orientation to ACC	

APPENDIX A - LIST OF COMMANDS
(continued)

Command Character	Command Syntax	Command Semantics	Paragraph
+	+	ACC = ACC + 1	
-	-	ACC = ACC - 1	
<number>	<number><command>	Iterates <command> by value of <number>	
=	=<name><command>	Defines user command	
	= # <name>	Defines user variable = ACC value	
?	? <then><else>	Random test	
[[<command>...]	Same as () except ACC saved & restored	
-	-	No-op	
^	^	Increment current iteration count	
CTRL-A	<option>	Audio select	
CTRL-C		Clear (remove) all user variables	
CTRL-D	<mode>	Select display mode	
CTRL-E	<rule>	Select edge rule	
CTRL-G	"<device>"	Get user definitions from device	
CTRL-L	<name>	Load user definitions from ROM	
CTRL-M	<mode>	Select operating mode	
CTRL-P	"<device>"	Put user definitions to device	
CTRL-R	<name>	Load & run user command from ROM	
CTRL-S	<option>	Command execution speed select	
CTRL-T	<option>	Select turtle representation	

APPENDIX B - ERROR STATUS CODES

- A - Operator Abort (BREAK key)
- D - Device name error
- F - User command definition area full
- I - Systems I/O error
- L - Load/Run argument undefined
- N - Nesting error (unmatched right bracket)
- O - Overlength command line input
- P - Incomplete (partial) line input
- R - Reserved name for user command
- S - Stack overflow
- U - Undefined user variable name used

APPENDIX D - PEN SELECTS (BY SCREEN MODE)

Mode	Pen Range	Color Registers Used
0	0 - 127	
1	0 - 127	
2	0 - 3	0 = background 1 = PF0 2 = PF1 3 = PF2
3	0 - 1	0 = background 1 = PF0
4	0 - 3	0 = background 1 = PF0 2 = PF1 3 = PF2
5	0 - 1	0 = background 1 = PF0
6	0 - 3	0 = background 1 = PF0 2 = PF1 3 = PF2
7	0 - 1	0 = PF2 1 = PF1 (luminescence only)

&0 = PF0
 &1 = PF1
 &2 = PF2
 &3 = PF3
 (&4 = background -- someday)

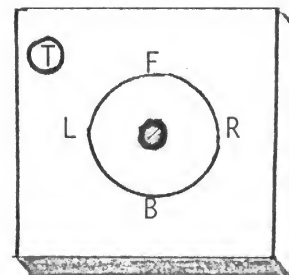
APPENDIX E - COLOR REGISTER VALUES

<u>Range</u>	<u>Mid-Bright</u>	<u>Color</u>
0 - 15	8	
16 - 31	24	
32 - 47	40	
48 - 63	56	
64 - 79	72	
80 - 95	88	
96 - 111	104	
112 - 127	120	
128 - 143	136	
144 - 159	152	
160 - 175	168	
176 - 191	184	
192 - 207	200	
208 - 223	216	
224 - 239	232	
240 - 255	248	

APPENDIX F - JOYSTICK TRIGGER TESTS

Select Code	Direction* Trigger	Device	Select Code	Direction Trigger	Device
A	F	JOYSTICK 0	U	T	POT 4
B	R	JOYSTICK 0	V	T	POT 5
C	B	JOYSTICK 0	W	T	POT 6
D	L	JOYSTICK 0	X	T	POT 7
E	F	JOYSTICK 1	Y	T	JOYSTICK 0
F	R	JOYSTICK 1	Z	T	JOYSTICK 1
G	B	JOYSTICK 1	[T	JOYSTICK 2
H	L	JOYSTICK 1	\	T	JOYSTICK 3
I	F	JOYSTICK 2			
J	R	JOYSTICK 2			
K	B	JOYSTICK 2			
L	L	JOYSTICK 2			
M	F	JOYSTICK 3			
N	R	JOYSTICK 3			
O	B	JOYSTICK 3			
P	L	JOYSTICK 3			
Q	T	POT 0			
R	T	POT 1			
S	T	POT 2			
T	T	POT 3			

* F = Forward
 R = Right
 B = Backward
 L = Left
 T = Trigger



APPENDIX G - ROM RESIDENT USER COMMANDS

(See p.15)

Load/Run Name	Content	Runs	Reference
1	Y = SIERPINSKI CURVE J = HILBERT CURVE	Y	See 4.8, 4.9
2	Y = TRINARY TREE J = SIMPLE SPIRAL	Y	See 4.10, 4.11
3	K = SUPER SPIRAL	K	See 4.12
4	Y = DRAW (CARTESIAN) J = DRAW (POLAR)	Y	
5	Y = WALL BANGER J = Breakout	Y	
6	J = HOLLYWOOD SQUARES	J	
7	Y = KOCH CURVE	Y	See 4.17
8	Y = THE ZAPPER	Y	FROM "ATARI WSFN" MANUAL
9	Y = DRAW	Y	
A	Y = POSIES	Y	
B	J = SUPERTURTLE	J	
C	COLORPOWER		
D	X = Magic Carpet	Y	

APPENDIX H - AUDIO SELECT OPTIONS

- Ø - audio off
 - A - LSB of horizontal position
 - B - LSB of vertical position
 - C - MSB of software stack pointer
 - D - Pot controller Ø
 - E - Pot controller 1
 - F - value in CHAR (current command)
 - G - (internal)- TEMP
 - H - (internal)- TEMP + 1
 - I - (internal)- COUNT
 - J - hardware stack pointer
 - K - MSB of horizontal position
 - L - MSB of vertical position
 - M - LSB of software stack pointer
 - N - (internal)- INPT +2
 - O - LSB of turtle accumulator
- } relative to upper left corner
- } relative to screen center