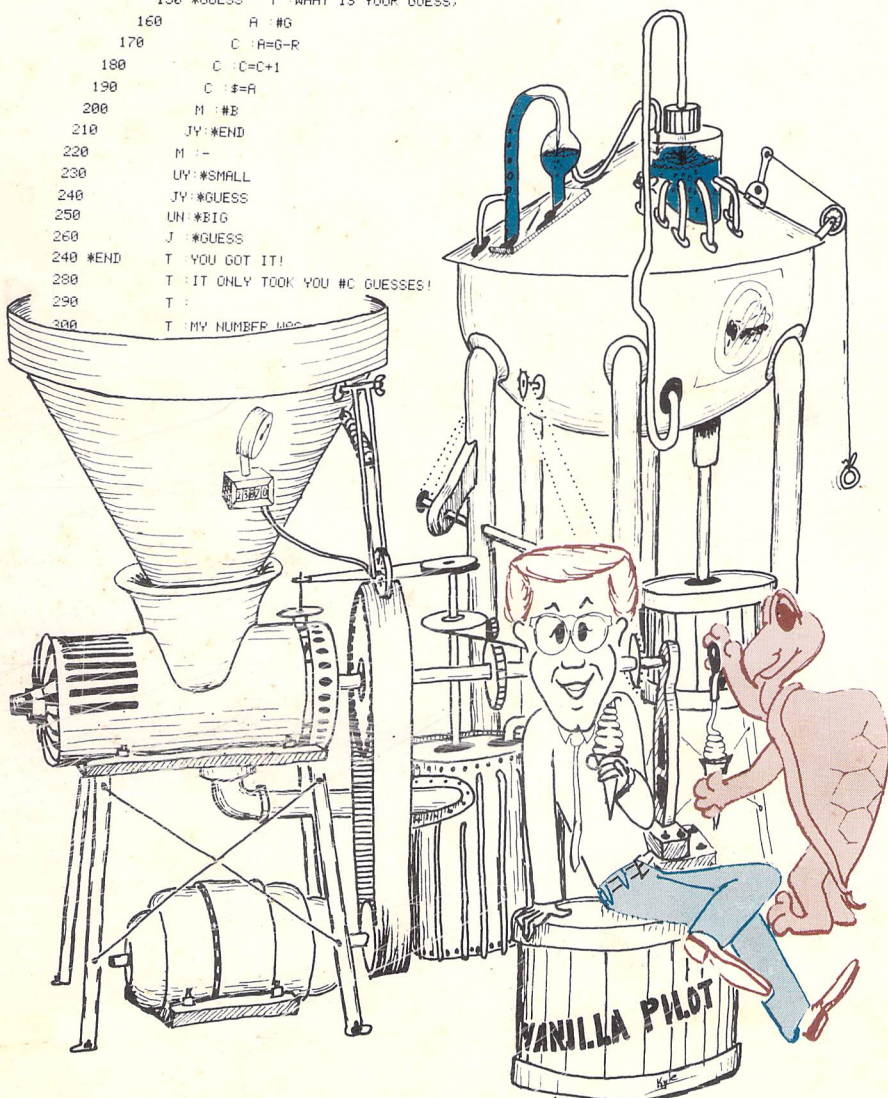


VANILLA PILOT

```

100      R : GUESS MY NUMBER
110      T : GUESS MY NUMBER BETWEEN 0 AND 99
120      C : C=0
130      C : E=0
140      N : F
150 *GUESS   T : WHAT IS YOUR GUESS
160      A : #G
170      C : A=G-R
180      C : C=C+1
190      C : E=A
200      M : #B
210      JY : #END
220      M : -
230      UY : #SMALL
240      JY : #GUESS
250      UN : #BIG
260      J : #GUESS
240 *END    T : YOU GOT IT!
280      T : IT ONLY TOOK YOU #C GUESSES!
290      T :
300      T : MY NUMBER WAS

```



Tamarack Software

VANILLA PILOT



Tamarack Software

Copyright 1982 by Tamarack Software, Inc. No part of this manual, except for brief passages in articles and reviews that refer to author and publisher, may be reproduced without written permission of the author.

WARRANTEE DISCLAIMER

The computer programs supplied with this manual are provided to you, the user, with no warrantee of any kind. Although due care has been taken to ensure the correct operation of this program, no representation about their fitness for any particular use or about the accuracy of the results. Tamarack Software, Inc., its distributors, retailers, or agents thus assume no responsibility and accept no consequential, incidental, or other liability arising from the use of these programs. Some states do not allow the exclusion or limitation or implied warranties or liability for incidental or consequential damages. So the above limitations may not apply to you.

HOW TO USE THIS MANUAL

The best way to learn how to program in VANILLA PILOT is to actually sit down with a computer and do the things you are reading about. Try the examples and write your own programs; then your knowledge will go beyond the theoretical to the practical.

The first eight chapters of the manual are written as a tutorial. Each of the various commands is highlighted in bold print. Then there is an explanation of what the command does and how to use it. Following the explanation are examples for you to try. The examples are illustrated with both program listings and video display screens, so you can see what should be happening as you work.

At the end of each chapter is a quiz to test your understanding of that chapter. There are questions testing your knowledge of the terms and commands and a program for you to write using the things you have learned. Review the chapter until you are comfortable with your knowledge of the terms and commands and of your ability to use them.

If you are already familiar with Pilot, Appendix A lists all the commands with a short explanation of each command and its format.

Remember HANDS ON FOR LEARNING will make you or your students proficient programmers of VANILLA PILOT.

VANILLA PILOT is a registered trademark of Tamarack Software, Inc. COMMODORE-64, VIC-20, CBM, PET are all trademarks of Commodore Business Machines, Inc.

ACKNOWLEDGEMENTS

The author would like to express his appreciation to the many people who helped make this product better. There are too many people to mention, but he would like to acknowledge the assistance of a few.

First, the encouragement and support of several people from Commodore Business Machines, Inc. who helped get this project started. And to Kevin Kyle for bringing life to the manual through his illustrations. Appreciation is also extended to Bill Hicks and Sharon Bush for allowing access to their classrooms. Their students gave us much insight into the approach to writing this manual. The thirty seven students in their classrooms helped us to find some more obscure programming errors.

Finally, thanks to the author's family who assisted in proof reading and bug catching. This includes his six year old daughter who, on her way to learning Vanilla Pilot, found several program errors everyone else overlooked.

TABLE OF CONTENTS

VANILLA PILOT

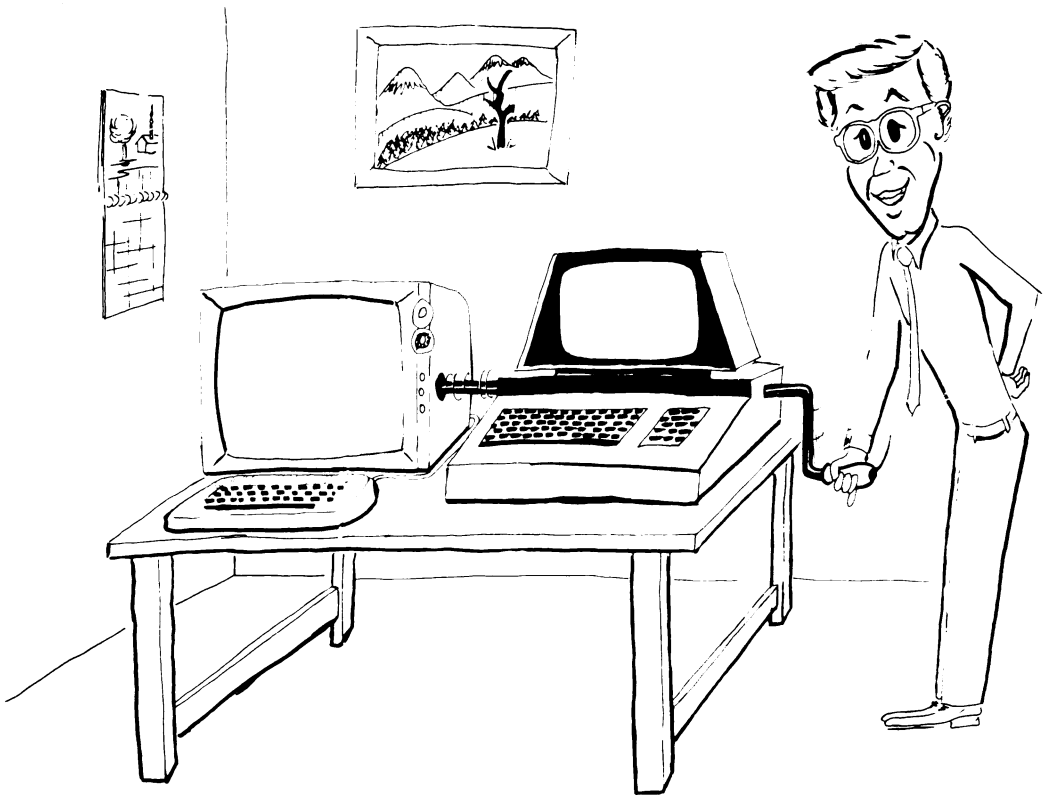
I. GETTING STARTED WITH PILOT	1
Clearing the Screen	4
What is a Computer Program	5
Entering your First Program	6
Some PILOT Editor Commands	7
The LIST Command	7
The NEW and UNNEW Commands	8
The AUTO Command	10
More PILOT Commands	11
The ACCEPT Command	11
The REMARK Command	13
The END Command	14
Quiz Yourself	14
II. DECISION MAKING PROGRAMS	15
The MATCH Command	16
Conditional Commands	18
The TYPE with Conditional Commands	18
More Editing Features	22
Deleting a Program Line	22
Deleting Program Sections	23
Inserting New Program Lines	23
Replacing a Program Line	24
The RENUMBER Command	25
Advance Renumbering	28
Quiz Yourself	29
III. BRANCHING	31
Statement Labels	32
The JUMP Command	32
The JUMP with Conditional Commands	34
Program Loading and Saving	35
Using a Cassette	35
Saving a Program on Cassette	35
Loading a Program from Tape	36
Using a Diskette	36
Saving a Program to Disk	37
Loading a Program from Diskette	38
Appending a Program	38
Checking the Error Channel	39
Disk Directory	39
Quiz Yourself	39

IV. NUMBERS AND VARIABLES	41
The COMPUTE Command	43
The RANDOM NUMBER Command	45
Debugging a VANILLA PILOT Program	45
The TRACE Command	46
The OFF Command	47
The Deferred Mode TRACE	47
The DUMP Command	47
Stopping and Re-Starting a Program	48
Quiz Yourself	49
V. MODULAR PROGRAMMING	51
The USE Command	52
The END Command	53
The USE with Conditional Command	54
The LList and PLIST commands	55
Quiz Yourself	57
VI. THE SCREEN AND SOUND COMMANDS	59
The SCREEN Command	60
The BEEP Command	66
Commodore-64 Version	66
VIC-20 Version	68
The CBM and PET computers	69
Quiz Yourself	70
VII. TURTLE GRAPHICS	71
The GRAPHICS Command	72
Quiz Yourself	80
VIII. SOME ADVANCED CONCEPTS	81
The PAUSE Command	82
The NAME Field	82
TYPE Reserved Characters	83
The JUMP Commands	83
The WAIT Command	84
Final Editor Commands	85
The FIND Command	85
The CHANGE Command	85
Quiz Yourself	86
Appendix A. PILOT REFERENCE MANUAL	87
PILOT Editor Commands	87
PILOT Interpreter Statements	93
Appendix B. PREPARING A DISKETTE FOR USE	105

Appendix C. PILOT ERROR MESSAGES	107
Editor Error Messages	107
Interpreter Error Messages	108
Turtle Graphics Error Messages	109
Appendix D. CONTROLLING THE JOYSTICK ON THE COMMODORE-64	
AND VIC-20	111
The JOYSTICK command	111
The FIRE BUTTON Command	112

I.

GETTING STARTED WITH PILOT



Chapter 1

There are several different versions of TAMARACK SOFTWARE'S PILOT. Follow the directions for your version of Vanilla Pilot to get started.

DIRECTIONS FOR LOADING

CBM-64 Diskette Version

Place diskette into disk drive.

Type: LOAD"LOADER*",8

When READY appears, type RUN

Press RETURN when directed.

CBM-64 Cassette Version

Place cassette into recorder.

Type: LOAD"VANILLA PILOT",1,1

Press the RETURN key.

When display returns, press CTRL key.

When ready appears, type: SYS32768

RETURN and you're ready to start.

VIC-20 Diskette Version

Insert 16k memory cartridge.

Place diskette into disk drive.

Type: LOAD"LOADER*",8

When READY appears, type Run

Press RETURN when directed.

VIC-20 Cassette Version

Insert 16K memory cartridge.

Place cassette into recorder.

Type: LOAD"VANILLA PILOT",1,1

Press the RETURN key.

When ready appears, type: SYS16384

RETURN and you're ready to start.

8032 and 4032 Diskette Versions

Place diskette into disk drive.

Press the SHIFT and RUN STOP keys together.

Press RETURN when directed.

8032 and 4032 Cassette Versions

Place cassette into recorder.

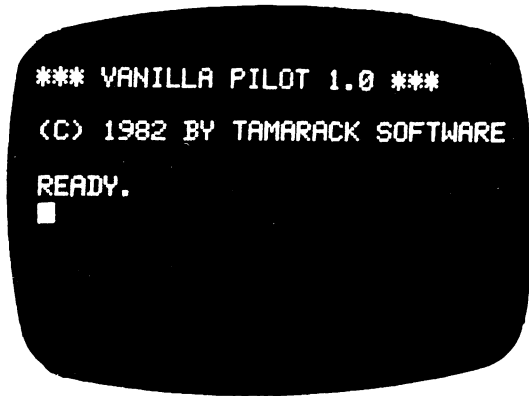
Type: LOAD"VANILLA PILOT"

Press the RETURN key.

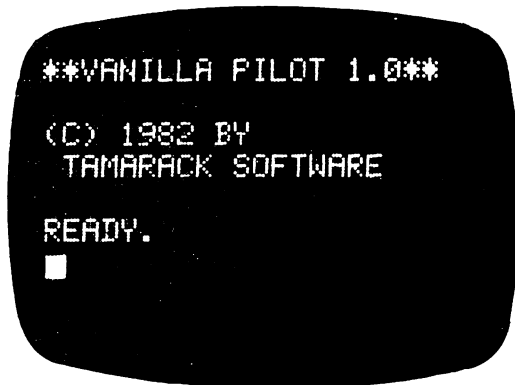
When ready appears type: SYS24576

RETURN and you're ready to start.

After starting Vanilla Pilot, your screen should look like this:



Or with the VIC-20, it should look like this:



In the manual the ■ represents the cursor. The cursor is a flashing square, whose job is to tell you where you are on the screen. It should now be below the R in READY. The word READY simply tells you that the computer is waiting for you to type something on the screen.

Whenever you type anything on the keyboard, the cursor will move one space to the right. Try it. Type: P I L O T. As you type PILOT, the cursor will move to the right. The cursor will now be just to the right of the T, waiting for further input, or typing.

```
*** VANILLA PILOT 1.0 ***  
(C) 1982 BY TAMARACK SOFTWARE  
READY.  
PILOT ■
```

Now press the key labeled **RETURN** . The job of the **RETURN** is to tell the computer to process the things you have typed. The computer does not know anything about what you have typed until the **RETURN** key is pressed. The computer will display:

```
*** VANILLA PILOT 1.0 ***  
(C) 1982 BY TAMARACK SOFTWARE  
READY.  
PILOT  
  
?SYNTAX ERROR  
READY.  
■
```

The computer responded with a ?SYNTAX ERROR because it didn't understand your input. When you type something the computer doesn't understand, it will display the words ?SYNTAX ERROR.

We will find out how to type some things the computer will understand in a later section.

CLEARING THE SCREEN

Now let's clear the screen. Clearing the screen means to erase everything on the screen. It is simple - hold down the **SHIFT** key and the **CLR/HOME** key together.

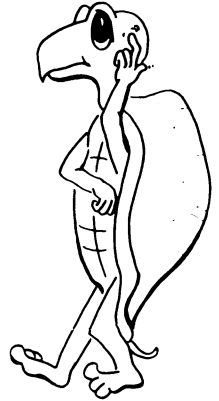
Try it!

Your screen is now blank except for the cursor waiting in the upper left corner. It is ready for your input.

Be sure to start each new function or activity by clearing the screen. If your computer is a VIC-20 or Commodore-64, NEVER use the **RESTORE** and **RUN/STOP** keys to clear the screen.

WHAT IS A COMPUTER PROGRAM?

A computer program is a series of instructions or commands that tell the computer to do a certain task or job. A program is made up of statements. A statement is a command or set of commands with a line number in front of it.



Line numbers help the computer tell the difference between immediate commands and program commands. A statement without a line number is an immediate command. When you press **RETURN**, the computer will do that command right then. A statement with a line number is a deferred or program command. The computer will not do that command until you type RUN. A computer program consists of a set of deferred statements.

A line number may be any whole number between 0 and 63999. Fractional line numbers are not allowed. If you use the same line number more than once, some of the statements will be lost.

Here is an example of a short program:

```
100 T:HELLO THERE!  
110 T:MY NAME IS JANE.  
120 T:MY COMPUTER IS CALLED  
130 T:SEYMOUR.  
140 E:  
■
```

This program has a set of lines. Each line starts with a line number and ends with a command. Line 100 contains a TYPE statement and will tell the computer to put, or type, everything following the colon (:) onto the screen. Lines 110, 120, and 130 are also TYPE statements. The E: in line 140 is an END statement. This tells the computer the program is finished.

ENTERING YOUR FIRST PROGRAM

Type **NEW** and press **RETURN**. Now type the lines of the above program on the computer. Type them exactly as they appear above, using only unshifted characters for your commands. Remember to press the **RETURN** key after each line.

OOPS!

Line 100 has a spelling error.

```
100 T:HELO THERE!
```

To correct errors you can either

- A. Retype the whole line with the error.
- B. Use the cursor control keys to correct the mistake.

Using the cursor control keys is usually a lot easier than retyping the whole line. Try it. Press the **SHIFT** key along with the key labeled **CRSR** and has arrows pointing up and down. The cursor will move up one line. On some computers it will keep moving up as long as you hold it down. Move the cursor to line 100. If you go up too far, then release both keys and press the same **CRSR** key again. This time do not press the **SHIFT** key. The cursor will move down.

Now that you have the cursor on the same line as the number 100 use the other **CRSR** key to move the cursor to the right. If you **SHIFT** with this key, the cursor will move left. Put the cursor on the letter O.

Press the **SHIFT** and the key marked **INST / DEL**. Wow! All of the letters, O THERE!, will jump to the right one space! Type the missing L and press the **RETURN** key.

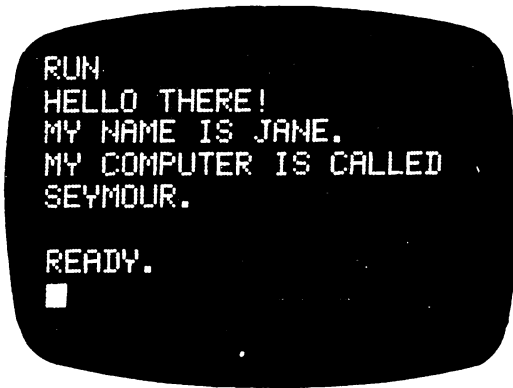
Now line 100 should look like this:

```
100 T:HELLO THERE!
```


You have just WRITTEN and EDITED your first program in VANILLA PILOT! How do you get the computer to follow these instructions? Press the **SHIFT** and **CLEAR/HOME** keys to clear the screen. Then type RUN and press the RETURN key.

Go ahead! Try it! Watch your program go!

Here is what you should see:



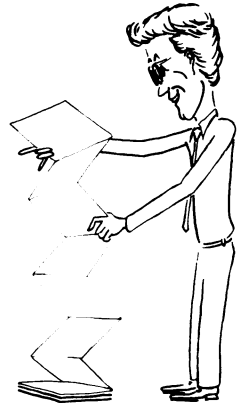
SOME PILOT EDITOR COMMANDS

An EXECUTIVE command is one which tells your computer what to do with your program. These commands do not have line numbers. In fact, if you use them with line numbers the computer might become confused and not know what to do with them.

The LIST command

The LIST command displays or lists the program statements currently in the computer's memory on the screen.

To display our example program clear the screen. Now type: LIST and press the **RETURN** key. The program statements will then be listed in order on the screen.



```
LIST
```

```
100 T:HELLO THERE!  
110 T:MY NAME IS JANE.  
120 T:MY COMPUTER IS CALLED  
130 T:SEYMOUR.  
140 E:
```

```
READY.  
■
```

Now you can check the program for mistakes, then change, add, or delete lines as necessary. The various ways to erase, replace, or insert lines will be talked about in a later chapter.

The LIST command tells the computer to list entire programs or portions of programs, depending on what you tell the computer to do. Here are some examples:

LIST	display the entire program.
LIST 100	display line 100.
LIST 130	display line 130.
LIST -120	display lines 0 to 120.
LIST 120-	display lines 120 to end of program.
LIST 110-130	display lines 110 to 130.

Try some of these examples on the sample program in memory.

When entering two numbers of lines to be listed, the first line number must be smaller than the second line number. If it is not, the computer will respond with a blank line. If you LIST a line number that doesn't exist, the computer also responds with a blank line.

The NEW and UNNEW commands

The NEW command erases the current program from the computer's memory. Be VERY careful when using the NEW command, it is easy to lose your program accidentally.

The UNNEW command will recover the program, if you accidentally type NEW. It will work, only if you have not placed any program lines in memory after the NEW command.

Try it.

Erase the program from memory. Type

NEW and press **RETURN**.

```
NEW
READY.
█
```

Try to list the program example again.

```
NEW
READY.
LIST
READY.
█
```

You have erased the program from memory.

Now let's test the UNNEW command. Type

UNNEW and press **RETURN**

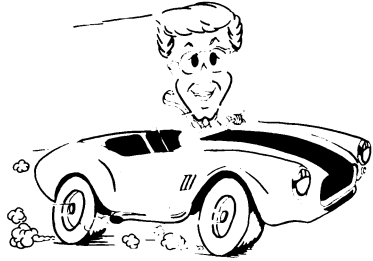
```
NEW
READY.
LIST
READY.
UNNEW
READY.
LIST
100 T:HELLO THERE!
110 T:MY NAME IS JANE.
120 T:MY COMPUTER IS CALLED
130 T:SEYMOUR.
140 E:
READY.
█
```

It works!

Always erase an old program from the computer's memory with the NEW command before entering a new program, otherwise, you will end up with a mixture of both programs.

The AUTO command

The AUTO command automatically assigns line numbers to program statements for you. As long as you enter the statements in proper order, this command replaces the task of sequential line numbering.



The AUTO command requires a number FOLLOWING it. This number tells the computer the increment between line numbers. Thus, AUTO 10 will tell the computer that the next line number is to be 10 units higher than the last number. The increment may be any number between 1 and 255.

Try this example. Enter the AUTO mode by typing: AUTO 10 and pressing the **RETURN** key.



Now you are ready to enter another program. In this program start with line number 100. Type: 100 T:ONE and press the **RETURN** key.

The screen will look like this:



When you pressed the **RETURN** key, the computer responded by typing the next line number and placing the cursor to the right of it. The computer is now ready for you to type in the next line of your program.

Once you've entered the entire program and wish to leave the AUTO mode, press the **RETURN** key without entering anything after the line number. Then type: AUTO and press the **RETURN** key. This will turn off the AUTO mode.

MORE PILOT COMMANDS

The ACCEPT command -- A:

The accept command will take input from the keyboard while the program is executing or running. When the computer comes to the ACCEPT command, it will stop and wait for you to enter something on the keyboard.

The format for the ACCEPT command is A:.

Type in the example program below:

```
100 T:I AM A COMPUTER
110 T:MY NAME IS SEYMOUR
120 T:WHAT IS YOUR NAME?
130 A:
140 T:I LIKE YOUR NAME!
150 T:HAVE A NICE DAY!
160 E:
```

What happens when you run the program:

```
RUN
```

```
I AM A COMPUTER  
MY NAME IS SEYMOUR.  
WHAT IS YOUR NAME?
```

The computer is waiting for you to type in your name. The question mark followed by an underline is the computer's signal for you to answer a question. You may type your name on the keyboard. If you make a mistake, use the **INST/DEL** key to remove the error, correct it, and then type the rest of your input.

When your answer is correct, press the **RETURN** key.

Here is the rest of the program:

```
RUN
```

```
I AM A COMPUTER  
MY NAME IS SEYMOUR.  
WHAT IS YOUR NAME?  
NAPOLEON  
I LIKE YOUR NAME!  
HAVE A NICE DAY!
```

```
READY.
```



NOTE: Always precede the **ACCEPT** command with a **TYPE** command to give the user a hint about what he should input.

Another NOTE: Use a semicolon (;) at the end of a **TYPE** statement to put the user input on the same line as the question. Remember if you have a question mark at the end of your question, remove it, as the **ACCEPT** command automatically prints a question mark.

For example, change line 120 in the above program to:

```
120 T:WHAT IS YOUR NAME;
```

Don't forget to press **RETURN** after typing the ;. Now run the program again



Now, as you can see, the answer will be on the same line as the question.

The REMARK command -- R:

The REMARK command will allow you to add titles or comments to your program to make it easier for you or others to understand. These notes can give more detail about what you are doing. The computer ignores REMARK commands, so you can insert them anywhere in your program.

The format for the remark command is

R:comments

If we insert some remark statements in the last program, we can see how they can be useful.

```
90 R:* * * THE NAME PROGRAM * * *
100 T:I AM A COMPUTER
110 T:MY NAME IS SEYMOUR
115 R:ASK FOR THE USER'S NAME
120 T:WHAT IS YOUR NAME?
130 A:
140 T:I LIKE YOUR NAME!
150 T:HAVE A NICE DAY!
160 E:
```

The END command -- E:

The END command is not necessarily the last line of the program. PILOT programs can be terminated, or ended, at any time with the END command. However, unless you use the end statement elsewhere, the last line must contain the END command.

Quiz Yourself

I. Match the command to its function.

- | | |
|----------|---|
| A. LIST | - Erases current program from computer's memory. |
| B. NEW | - Assigns line numbers to program statements. |
| C. AUTO | - Recovers the program. |
| D. UNNEW | - Displays on the screen the program statements currently in the computer's memory. |

II. Write a program using the TYPE, ACCEPT, REMARK, and END Commands.

II

DECISION-MAKING PROGRAMS



Chapter II

Vanilla Pilot programs can make decisions based on information input from the keyboard. The computer makes its decisions much the same way you do.

Making a decision involves accumulating data pertaining to the decision and then acting on that data. For example, you want to go to the park tomorrow for a picnic with your friends. You need to make plans today.

CAN WE HAVE A PICNIC TOMORROW?

Check the weather forecast.

If it is -

Sunny	Cloudy
Beautiful	Rainy
Warm	Cool
Clear	Hazy
Nice	Yucky

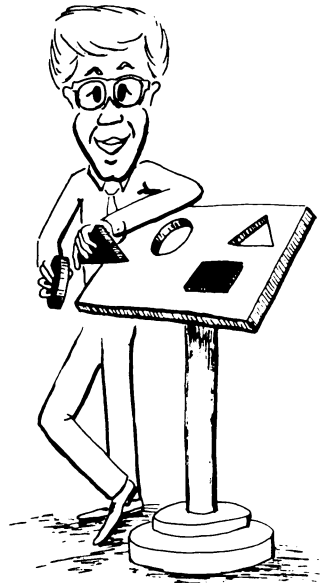
Then -

Have a nice	Stay home.
Picnic.	

There is an advantage to using pilot for writing decision-making programs. This advantage is that pilot was designed to be used when writing this sort of program. The program allows you to perform complex pattern-matching searches on the user input. Then you can make decisions based on the result of the match. This section shows you how to write pilot programs that can be useful or just plain FUN using pattern-matching.

The MATCH command -- M:

The MATCH command contains a list of patterns to be searched for and matched with user input information. Each pattern is called a STRING. A string is computer talk for a set of letters, numbers, or symbols. For example,



Have a nice day!	is a string.
Sally	is a string.
#\$%&'()@+:sx	is a string.
December 25	is a string.

The format for a match command is

M:string,string, . . . , string

There can be several strings in a MATCH command. If you have more than one, each must be separated by a comma.

```
M:HELP
M:HARD TEST,EASY QUIZ
M:13,14,15,16,17
M:CHAIR,TABLE,DESK
M:
```

NOTE: A MATCH command with no strings will match ANYTHING.

The MATCH command gets its information from the ACCEPT command. The ACCEPT command has a special function which retains anything you type on the keyboard. That information will be saved until the next ACCEPT command is used.

Anytime an ACCEPT is executed, the input is transferred to the answer field. The answer field is the area in the computer's memory where the user's responses are stored.

This form of the ACCEPT command is

A:\$

If you forget to put in the \$ sign, the computer will appear to accept the user input, but the information will not be saved in the answer field.

The MATCH command works by using what is called a 'sliding window' match. That is, each string in the MATCH command is scanned across the results of the last ACCEPT command, which is stored in the answer field, to see if there is a match.

```
M:AIR will match:  HOT AIR BALLOON
                   SPIRAL STAIRS
                   AIRPLANE
                   HAIRPIECE
```

However, M: AIR will NOT match SPIRAL STAIRS or HAIRPIECE. The space before the A of AIR will match with only the LEADING or beginning portions of a word.

CONDITIONAL COMMANDS

The MATCH command will search the answer field, which contains the results of the last ACCEPT command, for a matching string. The computer remembers whether or not it finds a match, then Vanilla Pilot can use this information to perform a CONDITIONAL command. A conditional command is one where you perform specific instructions based on the result of the last MATCH command. The result of the last match is stored in a location of the computer's memory called the Yes/No Flag.

Here are some examples of conditional commands:

TY:	TYPE if YES
TN:	TYPE if NO
AY:	ACCEPT if YES
AN:	ACCEPT if NO

YES/NO conditionals can be used with all the various commands.

The TYPE with conditional commands -- TY:, TN:

Your Pilot program can contain a TYPE statement used with a conditional command. This can be in the following formats:

TY:	message
TN:	message

When using the TY: command, if the last MATCH command found a match, then the TY command will be executed. If no match was found and there is a command with a N conditional, then it will be executed. Both the Y and N conditionals may be used following a single MATCH command, as long as they are not both in the same statement. If you use both, then ONLY one of the commands will be executed. The other, because its conditional does not match, will NOT be executed.

Here is a sample program:

```
100 R:GEOGRAPHY QUIZ
110 T:WHICH IS NOT A COUNTRY IN
```

```
120 T:NORTH AMERICA?
130 T:
140 T:CANADA MEXICO ENGLAND
150 T:
160 A:$
170 M:ENGLAND
180 T:
190 TY:THAT'S RIGHT!!!
200 TN:NO, ENGLAND IS IN EUROPE.
```

If the correct answer, ENGLAND, is entered when you RUN the program, the TY: command will be executed.

```
RUN
WHICH IS NOT A COUNTRY IN
NORTH AMERICA?

CANADA MEXICO ENGLAND

?ENGLAND

THAT'S RIGHT!!!

READY
■
```

If an incorrect answer is entered, the program will execute the TN: command to display:

```
RUN
WHICH IS NOT A COUNTRY IN
NORTH AMERICA?

CANADA MEXICO ENGLAND

?MEXICO

NO, ENGLAND IS IN EUROPE.

READY.
■
```

Now try another case. Type NEW and press the **RETURN** key.

```

100 R:WEATHER FORECAST
110 T:THE WEATHER FORECAST FOR TODAY
120 T:HAS A 100% PROBABILITY OF
130 T:PRECIPITATION.
140 T:
150 T:WHAT WILL WE SEE:
160 T:BLUE SKY CLOUDS RAIN SUNSHINE
170 A:$
180 T:
190 M:CLOUDS
200 TY:YES!! IT WILL BE CLOUDY.
210 M:RAIN
220 TY:RIGHT, PRECIPITATION MEANS RAIN.
230 M:CLOUDS,RAIN
240 TN:ARE YOU SURE ABOUT THAT?

```

Following are several RUNs of the program:

RUN

THE WEATHER FORECAST FOR TODAY
HAS A 100% PROBABILITY OF
PRECIPITATION

WHAT WILL WE SEE:
BLUE SKY CLOUDS RAIN SUNSHINE
?CLOUDS

YES!! IT WILL BE CLOUDY.

READY.

RUN

THE WEATHER FORECAST FOR TODAY
HAS A 100% PROBABILITY OF
PRECIPITATION.

WHAT WILL WE SEE:
BLUE SKY CLOUDS RAIN SUNSHINE
?SUNSHINE

ARE YOU SURE ABOUT THAT?

READY

```
RUN
```

```
THE WEATHER FORECAST FOR TODAY  
HAS A 100% PROBABILITY OF  
PRECIPITATION.
```

```
WHAT WILL WE SEE:  
BLUE SKY CLOUDS RAIN SUNSHINE  
?RAIN
```

```
RIGHT, PRECIPITATION MEANS RAIN.
```

```
READY.  
■
```

Notice that there are three different outputs, or computer responses, to what you typed in. The output depends on the input string.

What happens if you enter CLOUDS AND RAIN? Try it. Did your screen display:

```
RUN
```

```
THE WEATHER FORECAST FOR TODAY  
HAS A 100% PROBABILITY OF  
PRECIPITATION.
```

```
WHAT WILL WE SEE:  
BLUE SKY CLOUDS RAIN SUNSHINE  
?CLOUDS AND RAIN
```

```
YES!! IT WILL BE CLOUDY.  
RIGHT, PRECIPITATION MEANS RAIN.
```

```
READY.  
■
```

CLOUDS AND RAIN produces a match in both MATCH statements. So both TY: commands are executed.

Remember at the beginning of this chapter we were deciding whether to plan a picnic for tomorrow. You now have the tools to write a program to decide if you should go. Type: NEW and press the **RETURN** key. Enter the following program:

```
100 T:WHAT IS THE WEATHER FORECAST?  
110 A:$
```

What is the rest of the program? Try it! Your program should act like this:

```
RUN
```

```
WHAT IS THE WEATHER FORECAST?  
?SUNNY
```

```
HAVE A NICE PICNIC!
```

```
READY.  
■
```

```
RUN
```

```
WHAT IS THE WEATHER FORECAST?  
?CLOUDY
```

```
STAY HOME!
```

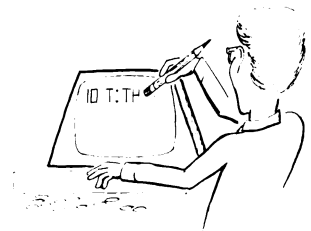
```
READY.  
■
```

MORE EDITING FEATURES

The Vanilla Pilot editor contains a number of features which make program entry and development easy.

Deleting a program line

When you wish to remove a program line from a Pilot program, it is very simple. You only need to type the line number of the line you wish to delete and press the **RETURN** key.



For example, enter the following program

```
100 T:TWINKLE,;  
110 T: TWINKLE;  
120 T: LITTLE;  
130 T: STAR
```

When you RUN this program, it should look like this

TWINKLE, TWINKLE LITTLE STAR

Now type: 120 and press **RETURN** . Then type: LIST and press **RETURN** .

```
120
LIST
```

```
100 T:TWINKLE;;
110 T: TWINKLE;
130 T: STAR
```

Line 120 is now permanently missing from the program. There is no way to recover it.

Deleting program sections

When you wish to remove an entire program section from a Pilot program, then use the DELETE command. The DELETE command has several formats

```
DELETE 100-200  Delete all lines 100 to 200.
DELETE -120      Delete all lines 0 to 120.
DELETE 500-      Delete all lines 500 to end.
DELETE 150       Delete only line 150.
```

Be very careful with both ways of deleting program lines. In either case, the lines you delete are PERMANENTLY gone from your program. There is no way to recover them.

Inserting new program lines

Adding a new program line to your program is quite simple. Whenever you type a program line, the computer will automatically enter it into the program at the correct position.

Let's try it! Remember our program

```
LIST

100 T:TWINKLE;;
110 T: TWINKLE;
130 T: STAR
```

Insert a new line

```
125 T: LITTLE;
```

List the program.

LIST

```
100 T:TWINKLE,;  
110 T: TWINKLE;  
125 T: LITTLE;  
130 T: STAR
```

Now we see that the new line, number 125, is in proper numerical sequence in the program. If we run this program, we see

TWINKLE, TWINKLE LITTLE STAR

Replacing a program line

Suppose you wish to delete a program line and insert another in its place. Try this by replacing line

```
125 T: LITTLE;
```

with

```
125 T: SMALL;
```

This can be done in two ways.

One, you can delete line 125 by typing 125 and pressing the **RETURN** key. Next, type the new line and press **RETURN**. Then list the program:

```
125  
125 T: SMALL;  
LIST  
  
100 T:TWINKLE,;  
110 T: TWINKLE;  
125 T: SMALL;  
130 T: STAR
```

You have replaced the old line 125 with the new line 125.

The second method is to type the new line 125 without first deleting the old line 125. The computer will then automatically replace the old line 125 with the new line 125. The old line is deleted because the computer doesn't allow duplicate line numbers.

Clear the screen and change line 125 back to the original

```
125 T: LITTLE;  
LIST
```

```
100 T: TWINKLE,;  
110 T: TWINKLE;  
125 T: LITTLE;  
130 T: STAR
```

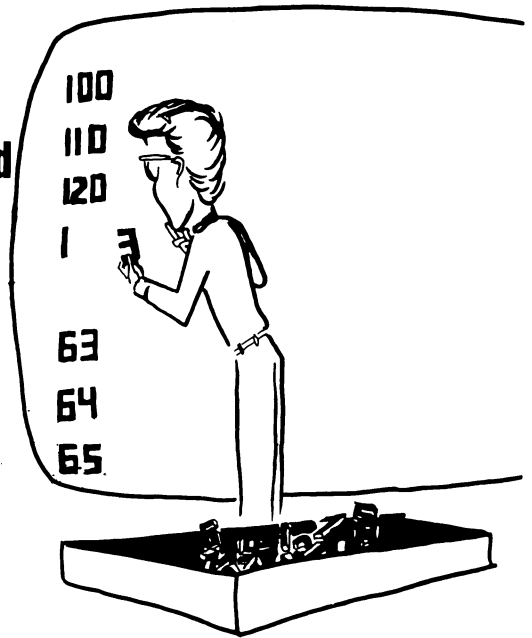
The RENUMBER command

Sometimes you might wish to insert a statement between two statements with consecutive line numbers. Vanilla Pilot has a RENUMBER command that will assign new line numbers to the program statements, thus giving you space to enter the new statement.

Type: NEW and press the **RETURN** key. Enter the following short program:

```
100 T: HELLO!  
101 T: GOOD BYE!  
102 E:
```

When you RUN this program, you will see:



Now you decide to enter a new line, so the program would display this:



```
HELLO!  
WHAT A NICE DAY!  
GOOD BYE!
```

```
READY.  
■
```

How would you enter the change? You could delete 101 and 102, then type in a new 101 and retype the old 101 and 102 as 102 and 103.

PHEW!

That is too much work! You could also use the RENUMBER command. The format for a simple RENUMBER command is: RENUMBER.

By itself, the RENUMBER command changes all of the line numbers in the program starting the new line numbers with line number 100 and counting by tens for as many lines as are in the program. This will create enough space to insert extra lines.

Type: RENUMBER and press the **RETURN** key. Now LIST the program.



```
LIST
```

```
100 T:HELLO!  
110 T:GOOD BYE!  
120 E
```

```
READY.  
■
```

You now have room to insert the extra line. Type: 105 T:WHAT A NICE DAY! and press the **RETURN** key. LIST the program again.

```
LIST
100 T:HELLO!
105 T:WHAT A NICE DAY!
110 T:GOOD BYE!
120 E:
READY.
■
```

Advanced renumbering

You can use the RENUMBER command to choose a starting line number and to work with increments other than 10. The format for this is

RENUMBER starting line number, new start, increment

The starting line number is the line where you wish to begin renumbering. The new start is the line number where you wish the renumbered portion to begin. The increment is the space between the lines.

The RENUMBER by itself is equivalent to typing RENUMBER 0,100,10

Acceptable formats for the RENUMBER command are:

<u>COMMAND</u>	<u>START #</u>	<u>NEW #</u>	<u>INCREMENT</u>
RENUMBER	0	100	10
RENUMBER 200	200	200	10
RENUMBER 200,500	200	500	10
RENUMBER 200,500,5	200	500	5

Remember that anytime you wish to specify any of the parameters, you must specify all those that would appear before that one.

Be careful! If you specify an increment that would take the highest line number above 63999, strange things can happen to your program line numbers.

Quiz Yourself

I. Fill in the blanks from the following list of words: MATCH, string, ACCEPT, conditional, DELETE, RENUMBER

1. A _____ is used to tie up a package and is also the name of the set of letters, numbers, and symbols to be searched for in a MATCH command.
2. When using the MATCH command, use the _____ command to transfer information from the user to the answer field.
3. You have written a program, and you need to add a new line between two consecutively numbered lines. Use the _____ command to assign new line numbers to the program statements.
4. Decision making is easy with yes/no _____ commands.

II. Did you write a program to decide whether or not to go on a picnic? If you didn't, write it now.

Clear the screen and change line 125 back to the original

```
125 T: LITTLE;  
LIST
```

```
100 T:TWINKLE;;  
110 T: TWINKLE;  
125 T: LITTLE;  
130 T: STAR
```

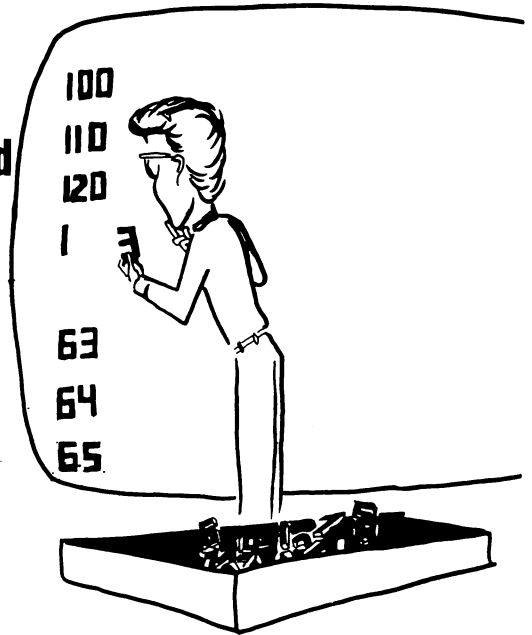
The RENUMBER command

Sometimes you might wish to insert a statement between two statements with consecutive line numbers. Vanilla Pilot has a RENUMBER command that will assign new line numbers to the program statements, thus giving you space to enter the new statement.

Type: NEW and press the **RETURN** key. Enter the following short program:

```
100 T:HELLO!  
101 T:GOOD BYE!  
102 E:
```

When you RUN this program, you will see:



Now you decide to enter a new line, so the program would display this:



```
HELLO!  
WHAT A NICE DAY!  
GOOD BYE!  
  
READY.  
█
```

How would you enter the change? You could delete 101 and 102, then type in a new 101 and retype the old 101 and 102 as 102 and 103.

PHEW!

That is too much work! You could also use the RENUMBER command. The format for a simple RENUMBER command is: RENUMBER.

By itself, the RENUMBER command changes all of the line numbers in the program starting the new line numbers with line number 100 and counting by tens for as many lines as are in the program. This will create enough space to insert extra lines.

Type: RENUMBER and press the **RETURN** key. Now LIST the program.



```
LIST  
  
100 T:HELLO!  
110 T:GOOD BYE!  
120 E  
  
READY.  
█
```


You now have room to insert the extra line. Type: 105 T:WHAT A NICE DAY! and press the **RETURN** key. LIST the program again.



Advanced renumbering

You can use the RENUMBER command to choose a starting line number and to work with increments other than 10. The format for this is

RENUMBER starting line number, new start, increment

The starting line number is the line where you wish to begin renumbering. The new start is the line number where you wish the renumbered portion to begin. The increment is the space between the lines.

The RENUMBER by itself is equivalent to typing RENUMBER 0,100,10

Acceptable formats for the RENUMBER command are:

<u>COMMAND</u>	<u>START #</u>	<u>NEW #</u>	<u>INCREMENT</u>
RENUMBER	0	100	10
RENUMBER 200	200	200	10
RENUMBER 200,500	200	500	10
RENUMBER 200,500,5	200	500	5

Remember that anytime you wish to specify any of the parameters, you must specify all those that would appear before that one.

Be careful! If you specify an increment that would take the highest line number above 63999, strange things can happen to your program line numbers.

Quiz Yourself

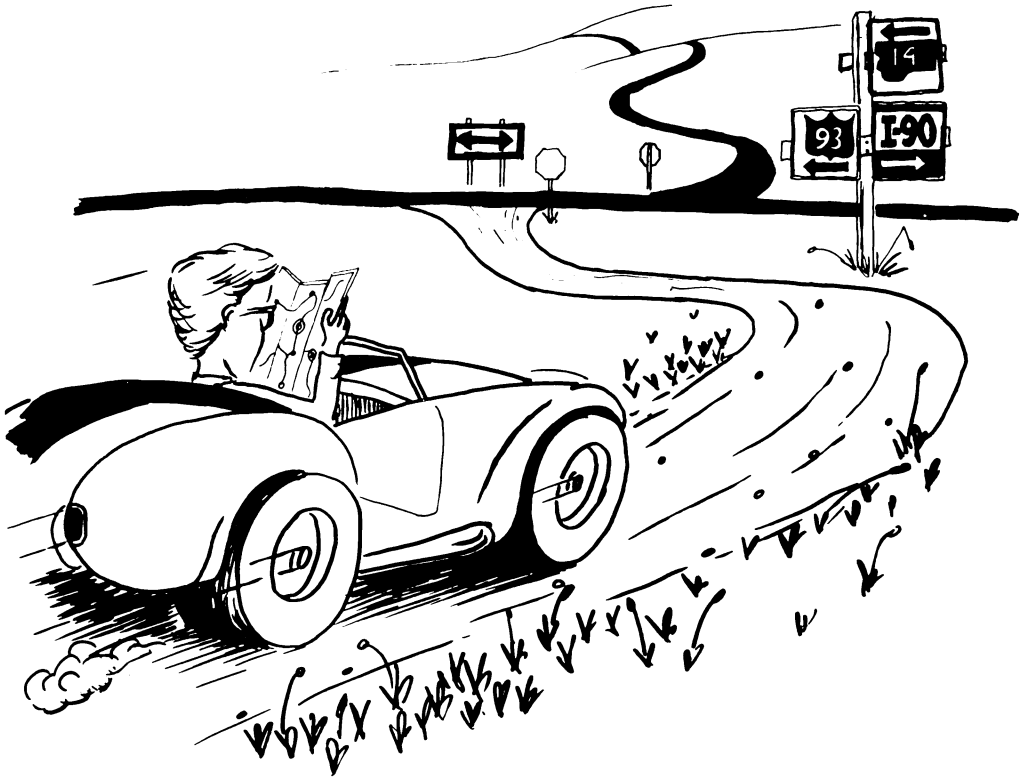
I. Fill in the blanks from the following list of words: MATCH, string, ACCEPT, conditional, DELETE, RENUMBER

1. A _____ is used to tie up a package and is also the name of the set of letters, numbers, and symbols to be searched for in a MATCH command.
2. When using the MATCH command, use the _____ command to transfer information from the user to the answer field.
3. You have written a program, and you need to add a new line between two consecutively numbered lines. Use the _____ command to assign new line numbers to the program statements.
4. Decision making is easy with yes/no _____ commands.

II. Did you write a program to decide whether or not to go on a picnic? If you didn't, write it now.

III

BRANCHING



All of the programs that you have been writing have executed in line number order. That is, a program line with a lower line number was executed before those with higher line numbers. This is called sequential program execution. In this chapter we will learn about the techniques for altering the flow of the program. Vanilla Pilot allows you to use something called a label. A label allows you to change the normal sequence of program execution.

Statement Labels

A statement label is a way of NAMING a Pilot statement line. It is different from a line number. Line numbers help to locate lines in relation to one another. Line numbers are used only when you are entering and editing a program. A label tells the computer "this line is special". Whenever you tell the computer to find a label, it will search for the label and transfer control to the line with the correct label.

A statement with a label has the format:

```
Line number *label name Pilot command
```

The label name must start with an asterisk followed by any combination of letters, numbers, or symbols. A space within the label is not allowed.

Some correct labels are:

```
100 *START T:HELLO THERE
130 *AGAIN A:
200 *TEST1.5 M:TYPE
```

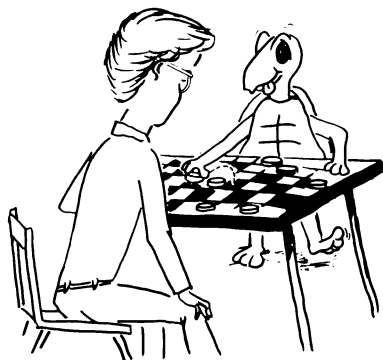
Some incorrect labels are:

```
150 *START HERE T:HELLO
    (space in label)
220 LABEL1 A:
    (no asterisk in front)
```

The JUMP command -- J:

The JUMP command tells the computer to break the usual sequential program execution and jump to a labeled statement.

The format for a JUMP command is:



J:label name

Some examples are:

```
120 J:*START
190 J:*AGAIN
200 J:*TEST1.5
```

Make sure that each JUMP command has a label to jump to. If you jump to a non-existent label, the program will stop and display the label name and say LABEL NOT FOUND.

If two or more statements have the same statement label, the JUMP command will jump to the line with the LOWER line number. Make sure that you do not have duplicate labels. Vanilla Pilot will search for the first few characters of a statement label. If an apparent match appears before the label you are actually looking for, the jump will be to the first label.

For example

```
200 *TEST1.5
.
.
.
300 J:*TEST
.
.
.
500 *TEST
```

If you used this JUMP command,

J:TEST

control would be transferred to the *TEST1.5 label even if you were actually searching for *TEST. The match is for ONLY the number of characters in the label following the JUMP statement. Be careful about the labels you use in the JUMP statement. You do not need to specify all of the characters in the destination label, but you do need to have enough of the label to find the correct label.

Try the following program

```
100 R:*** STATE PROGRAM ***
110 T:HERE IS A PROGRAM TO TEST
```

```

120 T:HOW MANY STATES YOU KNOW.
130 T:
140 *GUESS T:NAME A STATE;
150 A:
160 T:
170 J:GUESS
180 E:

```

When you RUN this program, it will continue forever asking you to name another state. There is no way for the program to reach the E: in line 180. You can stop the execution of a program at any time by pressing the **RUN/STOP** key.

The JUMP with conditional commands -- JY:, JN:

The JUMP command, following the MATCH command, can use the Yes/No conditionals to help in the decision making process. The MATCH command can instruct the JUMP command to go to the appropriate section of the program. The use of the Yes/NO conditionals with the JUMP statement is the same as with the TY: and TN: commands.

Now let's demonstrate this by modifying the program asking for the names of the states.

```

100 R*** STATE PROGRAM ***
110 T:HERE IS A PROGRAM TO TEST
120 T:HOW MANY STATES YOU KNOW.
130 T:
140 *GUESS T:NAME A STATE ;
150 A:
160 T:
163 T:WANT TO GUESS ANOTHER;
165 A:$
167 M:Y,YES,YEP,OK,FINE
170 JY:GUESS
180 E:

```

As you can see, we have added three lines 163-167. Type: RUN and press **RETURN**


```
RUN
HERE IS A PROGRAM TO TEST
HOW MANY STATES YOU KNOW.

NAME A STATE ?CALIFORNIA
WANT TO GUESS ANOTHER?YES
NAME A STATE ?MONTANA
WANT TO GUESS ANOTHER?NOPE
READY.
■
```



Program loading and saving

Using a Cassette

The Commodore (tm) Datasette will store Vanilla Pilot programs on a cassette tape and load them from that tape at a later time. Usually, you will store only one program per side of a tape, though more can be saved on each side, if you wish.

Saving a program on tape

Make sure that you have properly connected the recorder to the computer. If you are not sure about this, consult your computer manual. Now open the recorder lid by pressing the **STOP/EJECT** key. Insert a cassette tape. If this is the first program on the cassette, be sure that the tape has been advanced past the leader. Make a note of the tape counter setting.

To save the program onto the tape, type

```
CSAVE "filename"
```

The "filename" is the name you have given to the program. When you press the **RETURN** key, the computer will ask that

the **PLAY** and **RECORD** keys be pressed simultaneously. The recorder will run for a short while, then the word **READY.** will appear on the screen. The program has now been saved. If you are planning to use the tape for additional programs, make a note of the tape counter so you know where to start the next program.

You can protect the tape from having another program saved over the one already there. That is, you can **WRITE PROTECT** your tape. To do this, you must break out the small tabs on the rear of the tape. The proper one to break out is the one on the left, when the side to be write protected is facing up and the tape opening is facing toward you.

If you decide to reuse a write-protected tape, place a small piece of cellophane tape over the write-protect opening.

Loading a program from tape

Make sure that you have properly connected the recorder to the computer. If you are not sure about this, consult your computer manual. Now open the recorder lid by pressing the **STOP/EJECT** key. Insert the cassette tape with the program you wish to load and make sure it is fully rewound. Reset the tape counter to 000. Now advance the tape to the start of the program using the tape counter settings you wrote down when the program was saved.

To load the program from the tape into the computer memory type

`CLOAD "filename"`

The "filename" is the name of the program that you want to load. You do not need to type the name of the program if you are sure that the next program on the tape is the one you wish to load. Merely type `CLOAD` and the computer will ask that you press the **PLAY** key. The recorder will run for a short while and the word **READY.** will appear. Now you can either **RUN** the program or edit it.

Using a diskette

Programs can be saved onto diskettes using a disk drive unit. A disk drive unit can store many, many programs on it. The programs can be loaded at a later time.



Saving the program to disk

Make sure that the disk drive is properly connected to the computer. If you are not sure about this, consult your computer manual. Turn on the drive unit and place a diskette into the drive. If your diskette has never been used before, then you must format it before you save your program. See Appendix B for instructions on this.

To save a program on diskette, type

```
SAVE "dr:filename"
```

The dr is the drive number you wish to use for program storage. It is either 0 or 1. The "filename" is the name you wish to give to your program.

You can also save entire sections or modules from your program. If you wish to do this, you must specify the program lines to be saved following the program name.

```
SAVE "dr:filename",x-y
```

The x is the first program line to be saved, and the y is the last program line to be saved. Some examples are:

SAVE "0:STATES"	Save all of the STATES program.
SAVE "1:QUESTION",-300	Save lines 0 to 300
SAVE "0:COMPARE",550-800	Save lines 550 to 800

When you press the **RETURN** key, the active light will come on on the appropriate drive and in a few seconds the word READY. will appear. When this occurs, the program has been saved. The save a module option is not available to cassette users.

A diskette can be Write protected. That is, you can prevent any further programs from being saved on the diskette. If you hold the diskette with the label right-side-up, you will see a small notch on the right of the diskette. If you cover this notch with a small piece of cellophane tape, the diskette is write protected.

When you attempt to save a program on this diskette, the computer will refuse to write on it. To write on the diskette simply remove the tape.

Loading a program from diskette

Make sure that the disk drive is properly connected to the computer. If you are not sure about this, consult your computer manual. Turn on the drive unit. Now place a diskette into the drive. Type

```
LOAD "dr:Filename"
```

The dr is the drive number of the disk drive into which you placed your diskette. It is either 0 or 1. If you leave out the drive number and colon (:), the computer will search both drives of a dual drive for the program. The "filename" is the name of the program you wish to load. When you press the **RETURN** key, the drive will run for a few seconds and the program will be loaded.

When the READY. appears you can either EDIT, LIST, or RUN the program in memory.

If you wish to RUN the program after LOADING it from disk, you can do this in one step by typing

```
RUN "dr: FILENAME"
```

This acts just like a LOAD from disk, and then typing RUN from the keyboard.

Appending a program

Usually a LOAD will write the program over whatever is in memory. However, if you use the form

```
LOAD "dr:filename",x
```

the computer will LOAD the program called "filename" and begin it at line x in memory. It will be added on to the current program in memory beginning at line x. If any program lines are in memory at or following line x, they will be over-written by the new lines.

```
LOAD "0:TESTER"    load the file TESTER
LOAD "1:GUESS",2200 load Guess beginning at
2200
```

This function is not available with the CLOAD command.

Checking the Error Channel

Sometimes, when using the disk drive, the red light will flash off and on (with the 1540/1541 and 2031 disk drives) or will stay on (with the 4040, 8050 and 8250 disk drives). The red light indicates there is a disk error. To check to find out what that error is type ERROR on the Commodore-64 and VIC-20, and type ?DS\$ on all the other machines.

Disk Directory

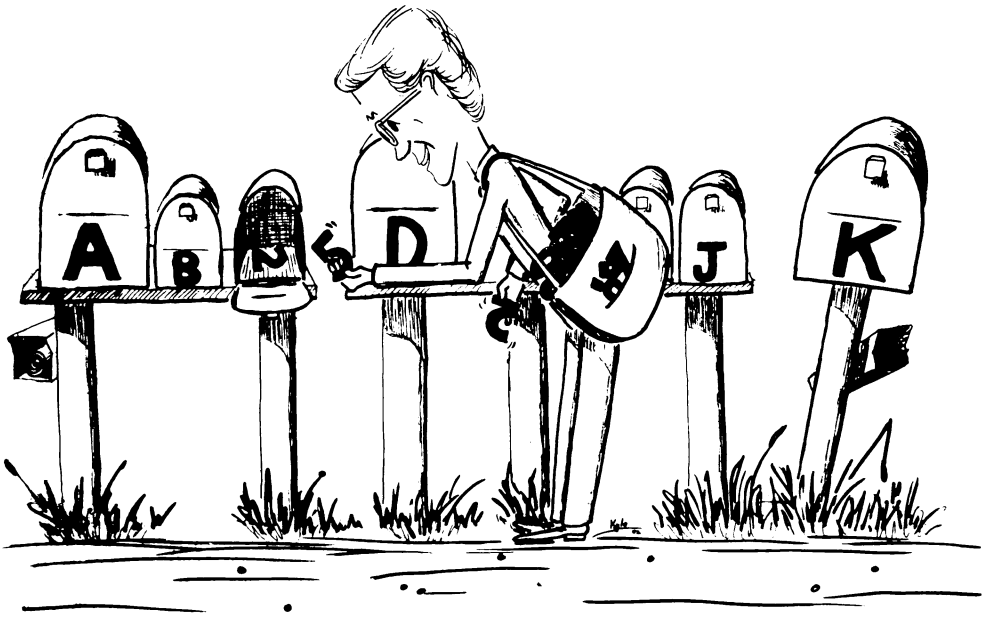
After you have saved a number of programs onto a diskette, you might want to see a list of all the programs on that diskette. To do this type the word Directory, and a list will be displayed on the screen.

Quiz Yourself

- I. Choose the best answer or answers for the following multiple choice questions.
1. The JUMP Command tells the computer to jump
 - A. In a lake.
 - B. To the beginning of the program.
 - C. To a labeled statement.
 - D. To the preceding line.
 2. To find out what programs are saved on a diskette, type
 - A. Save.
 - B. Directory.
 - C. List.
 - D. What's on the diskette, please?
 3. A small piece of cellophane tape is useful for
 - A. Displaying an amusing cartoon on the wall.
 - B. Taping your neighbors mouth closed.
 - C. Write-protecting your diskette so that no more programs can be save on it.
 - D. Unwrite-protecting your cassette so that you can save more programs on it.
- II. Write a program using statement labels, JUMP commands and JUMP commands with conditionals statements. Save your program on Datasette, or disk, or both.

IV.

NUMBERS AND VARIABLES



Chapter IV

This section is about numbers and how to use them in Vanilla Pilot. Numbers can be present in two forms. One form of number is an integer or a constant. That is, a whole number whose value is specified in the program. The other form is a variable. A variable is subject to change either by the user or by calculations in the program.

Examples of integers are

1234	These are all
6879	fixed numbers
882	whose value will
-309	NOT change.

These values will stay the same all through the program.

A variable is a number that is subject to change during the program operation. Often the programmer will not know its value while writing the program. Below are some examples.

X=Y+4	X and Y are variables.
The weather	Is a variable. It changes all the time.
The date	Is a variable. It changes every day.

Think of a variable as a mail box. You generally know where it is, but you don't know what it will contain. Like the postman, the computer can have several boxes to fill with numbers.

The computer has 26 variables for you to use. When using variables, give them names of the letters of the alphabet. The variables can contain only the numbers between -999 and 999. If you try to use a number outside this range, Vanilla Pilot will give you an error message

VALUE < -999
VALUE > 999

Also, use only WHOLE numbers in this range. Fractional numbers are not allowed.

The COMPUTE Command -- C:

The COMPUTE command will allow you to evaluate a numeric expression and assign its value to a numeric variable. The format for COMPUTE is:

C:numeric variable=numeric
expression

All calculations must be done with the COMPUTE command. Some examples of valid COMPUTE commands are:

C:X=X+5
C:Y=27
C:A=X-3+Y



The COMPUTE command can do only addition and subtraction. Here is an example program.:

```
100 R:CALCULATOR
110 T:THIS IS AN ADDING MACHINE PROGRAM
120 T:TYPE -999 TO SHOW THE SUM.
130 T:
140 C:S=0
150 *ADDIT T:WHAT IS THE NEXT NUMBER?
160 A:#N
170 T:
180 M:-999
190 JY:*END
200 C:S=S+N
210 J:*ADDIT
220 *END T:THE SUM IS #S.
230 E:
```

Besides the COMPUTE statements, this program has several other features that we haven't seen before. First, is line 160 A:#N. This statement will ask for a numeric value and store it in variable #N. Until now we have seen that the ACCEPT statement will display a ? to prompt you that it is expecting an input from you. This form of the ACCEPT statement will use a # sign. This is to show that you can only enter a number.

Next, look at line 220 *END T:THE SUM IS #S. The #S (the S is a variable) in a TYPE statement will show the value of the variable following the # sign.

Now let's RUN this program:

```
RUN
THIS IS AN ADDING MACHINE PROGRAM
TYPE -999 TO SHOW THE SUM.

WHAT IS THE NEXT NUMBER?
#34
WHAT IS THE NEXT NUMBER?
#45
WHAT IS THE NEXT NUMBER?
#-999
THE SUM IS 79.

READY.
■
```

Now let us RUN this program again

```
RUN
THIS IS AN ADDING MACHINE PROGRAM
TYPE -999 TO SHOW THE SUM.

WHAT IS THE NEXT NUMBER?
#684
WHAT IS THE NEXT NUMBER?
#438
ERROR - C:S=S+N
OVERFLOW IN CALCULATION

READY.
■
```

The error message here shows that the result of the addition exceeded 999 and the program was stopped. As Vanilla Pilot uses only numbers between -999 and 999, the same error message would appear if the number was lower than -999.

There is one special form of the COMPUTE command

$C:\$ = \text{numeric expression}$

This will transfer the result of the numeric expression into the ANSWER field. Since the MATCH command always compares with the contents of the ANSWER field, this command format permits the MATCH command to work on numbers. Study the following program.

100 T:THIS PROGRAM WILL PRINT

```

100 T:THIS PROGRAM WILL PRINT
110 T:THE WORD HELLO 17 TIMES.
120 T:
130 C:A=0
140 C:B=17
150 *AGAIN T:HELLO
160 C:A=A+1
170 C:$=A
180 M:#B
190 JN:*AGAIN
200 T:
210 T:THAT'S ALL!
220 E:

```

The above program uses this special COMPUTE feature. Line 180 has is a MATCH command using a numeric variable. This MATCH command will match the exact numeric values. If you were to write 180 M:17, then, because of the sliding window match described in chapter II, the program would print HELLO only once because 17 has the number 1 in it. As written, with 180 M:#B, the computer will print HELLO 17 times because this version of the MATCH command is for exact matching if numbers.

Try it!

The RANDOM NUMBER Command -- N:

A random number is any integer number between 0 and 99 generated by the computer. The format of this command is

N:x

where x can be any numeric variable between A and Z.

Debugging a VANILLA PILOT Program

As you begin to write longer programs in Vanilla Pilot, you will almost certainly come up with some program errors or BUGS!

UGH!!

Sometimes these bugs are very easy to spot and exterminate. Others are quite



difficult to find. Vanilla Pilot has some commands, both immediate and deferred, to make this easy.

The TRACE Command

This very versatile command will allow you to watch a Pilot program as it is running. It is an excellent way to check out a program - you can watch what the program is doing and spot where it gets into trouble. The TRACE command is also an excellent teaching tool. A beginner can watch exactly what the program is doing.

To start the TRACE type:

TRACE and press the **RETURN** key.

When you are ready to watch the program, type: RUN and press the **RETURN** key.

The program line currently being executed will appear with dark characters on a light background at the top of the screen on the CBM and PET machines. On the Commodore-64 and VIC-20 machines the color of the TRACE characters will depend upon the color of the current programs lines that are being executed. The program will execute exactly as it normally would, except much SLOWER. The lines at the top of the screen will change about once every second.

You can slow this down by pressing the **SHIFT** key (the **CONTROL** key for the Commodore-64). The TRACE will then stop and wait for you to press the **SHIFT** (or **CONTROL** on the 64) key again. When you press the **SHIFT** key, the TRACE will display the next program line. It can be a little tricky to exit this step mode. Usually pressing the **SHIFT** key twice in very rapid succession will do the trick.

The trace can be speeded up to about a fourth as fast as the program would normally run by pressing the **OFF/RVS** key (the equals key on the VIC-20). When you release this key, the TRACE will go back to the one line per second mode.

Vanilla Pilot does the TRACE in the following fashion. It will first print the line to be executed. Then execute the line. Next, it checks for the **SHIFT** or **OFF/RVS** keys or will wait for a second. If you have a Pilot statement that affects the first two lines (one line on an 80 column screen or four lines on the VIC-20) of the screen, it may affect this area and cover or erase the trace. Don't worry, this is quite normal. The trace will appear when the next line is executed.

The OFF Command

When the TRACE through a program is finished, type OFF and press the **RETURN** key, the TRACE function is then turned off. The TRACE will continue in effect until this command is issued or the computer is switched off.

The Deferred Mode TRACE -- ! :

Sometimes you will know the general area of the program where your BUG is occurring. Using the Deferred Mode TRACE, the program will execute normally until it reaches that specially marked section, then will trace through that section only. The Deferred Mode TRACE functions exactly like the immediate mode TRACE.

The format for the Deferred Mode TRACE is

! :

Locate the problem area in your program. Immediately before that area, enter a program line containing an !:. After the suspected area, enter another program line containing an !:. The first !: turns on the Deferred Mode TRACE; the second !: turns it off.

The DUMP Command



This command will display a formatted listing of the contents of the answer field, name field, yes/no flag and the contents of all 26 numeric variables. Simply type: DUMP and press the **RETURN** key.

A typical screen display might look like this

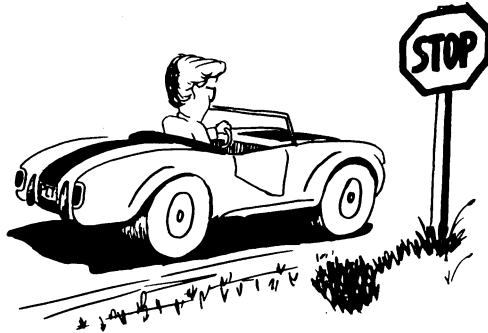
```
ANSWER FIELD -  
THIS IS A TEST  
NAME FIELD -  
DAVID  
Y/N FLAG - N
```

NUMERIC VARIABLES

A = 0	B = 13
C = 941	D = 0
E = 0	F = -38
G = 0	H = 0
I = 5	J = 0
K = 0	L = 0
M = 0	N = 0
O = 0	P = 0
Q = 3	R = 0
S = 0	T = 0
U = 0	V = 0
W = 0	X = 31
Y = 28	Z = 0

```
READY.  
■
```

Stopping and Re-starting a Program



There are times when you might wish to stop a Vanilla Pilot program and begin again from that same point. You can stop the program at any point by pressing the **RUN/STOP** key. However, this does not always stop the program at a predictable point.

You can insert a H: command on a line in the area where you wish to stop. This will halt the execution of the program and return you to the Vanilla Pilot editor. You can then examine variables, check the program code, etc.

After either using the **RUN/STOP** key or the H: command, you can type CONTINUE and press the **RETURN** key to resume execution of the program. You can continue the program execution unless you have done one of the following things:

1. Edited a program line.
2. Pressed the **RUN/STOP** key in an ACCEPT command.
3. LOAded or CLOAded another program.
4. CSAVEEd the program.
5. DELETEEd a portion of the program.
6. Used the CHANGE command.

If you have done any of these things, then you cannot continue with the program execution. The computer will respond with

CAN'T CONTINUE

and refuse to continue with the program.

Quiz Yourself

- I. Fill in the blank with the correct answer from the following list: Variable, COMPUTE, RANDOM NUMBER, TRACE, OFF, and DUMP.
1. The _____ command lets you follow step by step what the program is doing.
 2. To let the computer choose a number between 0 and 99 use the _____ command.
 3. A _____ is a number that may be changed either by the user or a calculation.
- II. Write a program to calculate how much money you will have left after paying for all your necessities for a week. Round off your numbers to the nearest dollar. Use the TRACE and the DEFERRED TRACE to see how your program runs or to debug your program.

V.

MODULAR PROGRAMMING



Chapter V

As you become a more experienced programmer, you will start writing longer, more complicated programs. You will find that many groups of program statements are used repeatedly within the same program. These statements could be repeated as needed, but this can be very tedious. An accomplished programmer will group these statements that are to be used more than once then branch to them as they are needed. A group of statements like this is called a **MODULE**. Modules, also called subroutines, are like miniature programs within a big program.

The use of modules is a powerful technique to streamline your Vanilla Pilot programs. Using modules allows you to make your programs easier to write, understand and modify. You can also develop a set of modules and reuse them in many programs. Vanilla Pilot programs can then be made shorter, thus, you can do more things in a single program.

The beginning of a module is marked by a statement label. This is followed by a set of program lines. When the module task is done, then an **END** command is placed on the last line of the module. This **END** command exits the module and returns to the statement following the line which called the module.

Here is a sample module:

```
900 *NUMBER T:PLEASE ENTER A NUMBER
910 T:BETWEEN 0 AND 999;
920 A:#N
930 E:
```

Every time you go to this module the computer will ask for a number between 0 and 999. This module, ***NUMBER**, could be called whenever you wish to enter a number into the program. At the end of the module, program execution returns to the main program.

The **USE** command -- **U:**

A module is called with the **USE** command. The format is

U:label

This command is always part of the main program. It is not part of a module, unless you wish one module to call another module.

The **USE** command is much like the **JUMP** command in that both will search for the line with the appropriate label. The differ-

ence between them is that the USE remembers the location of the line immediately following the USE command. When an END command is found the program does not end, rather the program returns to the main program immediately following the USE command.

HINT: A module can be placed anywhere in the program; even after the program END command. Vanilla Pilot will find the module, if it exists. However, you should place all of your modules at the beginning of your program so the program will run faster. The computer starts searching at the beginning of the program for a label.

The END command -- E:

THE END COMMAND MUST BE THE LAST STATEMENT OF A MODULE.

The format for the END command is

E:

Earlier we learned that the END command ends the program. The same END command used in a module will end that module and signal the computer to return to the main program. The END within a module does not end the entire program. The computer knows the difference between a module END and a program END.

A module can be nested. This means that one module can call another module. This in turn can call a third module and so forth. Vanilla Pilot allows up to seven nested modules in each series of nested modules. Whenever you nest a module, the computer will remember the correct line to return to when it sees an END command.



When nesting modules, it is easy to make programming errors. Errors like neglecting to end a module with an END command, or calling more than seven nested modules. If you happen to do so, this is the place where the TRACE (or the ! :) command will be a real value to you.

The USE with Conditional -- UY:, UN:

A module can be called conditionally by changing the USE command to include the Y or N conditionals. You have used these before with the T:, E:, J:, etc. commands. Look at the following example:

```
100 R:**** MENU PROGRAM ****
110 T:GUESS MY NUMBER
120 T:NIM
130 T:
140 T: WHICH GAME?
150 A:$
160 M:GUESS
170 UY:*NUMBER
180 UN:*NIM
190 E:
200 *NUMBER T:
210 T:HERE ARE THE RULES.
.
.
.
740 E:
750 *NIM T:
760 T:HERE ARE THE RULES.
.
.
.
990 E:
```

If GUESS or GUESS MY NUMBER is entered, the module *NUMBER is called. If anything else is entered, the program will call the *NIM module.

BEWARE - If you leave the module *NUMBER with the yes/no flag set to no, the computer will then call the *NIM module.

Here is a simple program to guess the computer's secret number. It is a more complex program than any we have looked at before. Try it!

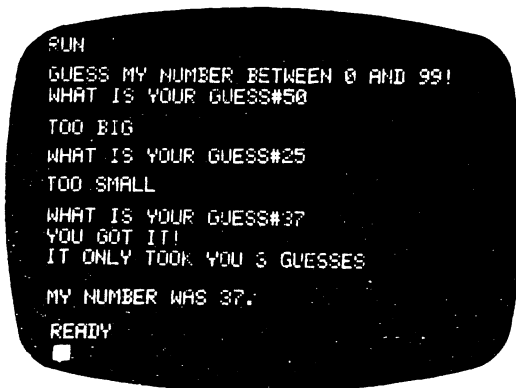
```
100 R:GUESS MY NUMBER
110 T:GUESS MY NUMBER BETWEEN 0 AND 99!
120 C:C=0
130 C:B=0
140 N:R
150 *GUESS T:WHAT IS YOUR GUESS;
160 A:#G
```

```

170 C:A=G-R
180 C:C=C+1
190 C:$=A
200 M:#B
210 JY:*END
220 M:-
230 UY:*SMALL
240 JY:*GUESS
250 UN:*BIG
260 J:*GUESS
270 *END T:YOU GOT IT!
280 T:IT ONLY TOOK YOU #C GUESSES!
290 T:
300 T:MY NUMBER WAS #R.
310 E:
320 *SMALL T:
330 T:TOO SMALL!
340 T:
350 E:
360 *BIG T:
370 T:TOO BIG.
380 T:
390 E:

```

Enter the program and run it. It should look like this:



```

RUN
GUESS MY NUMBER BETWEEN 0 AND 99!
WHAT IS YOUR GUESS#50
TOO BIG
WHAT IS YOUR GUESS#25
TOO SMALL
WHAT IS YOUR GUESS#37
YOU GOT IT!
IT ONLY TOOK YOU 3 GUESSES
MY NUMBER WAS 37.
READY

```

Study this program carefully. It contains a number of the features that we have studied in Vanilla Pilot so far.

The LLIST and PLIST Commands

These two commands will assist you in getting a listing of a program in a more readable format. LLIST will list a formatted listing on the screen while PLIST will list to printer device #4.

If we LLIST the program we just wrote, you can immediately see that it is easier to read.

LLIST

```
100      R:GUESS MY NUMBER
110      T:GUESS MY NUMBER BETWEEN 0 AND 99!
120      C:C=0
130      C:B=0
140      N:R
150 *GUESS T:WHAT IS YOUR GUESS;
160      A:#G
170      C:A=G-R
180      C:C=C+1
190      C:$=A
200      M:#B
210      JY:#END
220      M:-
230      UY:*SMALL
240      JY:*GUESS
250      UN:*BIG
260      J:*GUESS
270 *END  T:YOU GOT IT!
280      T:IT ONLY TOOK YOU #C GUESSES!
290      T:
300      T:MY NUMBER WAS #R.
310      E:
320 *SMALL T:
330      T:TOO SMALL!
340      T:
350      E:
360 *BIG  T:
370      T:TOO BIG.
380      T:
390      E:
```

Compare this listing with the previous one. Here it is much easier to see both the labels and the conditionals attached to the commands.

Quiz Yourself

I. Choose the best answer or answers for the following multiple choice questions.

1. Another word for module is

- A. Program.
- B. Subroutine.
- C. Statement.
- D. Command.

2. To print your program on a printer, type

- A. LIST.
- B. PLIST.
- C. Print.
- D. LLIST.

3. Use the END Command, E:,

- A. At the end of your program.
- B. When the screen is full.
- C. At the end of a module.
- D. At the end of each line.

II. Write a program using at least one module, or subroutine.

SECRET

TO THE SECRETARY OF THE ARMY
WASHINGTON, D. C.

FROM THE SECRETARY OF THE ARMY

SUBJECT: [Illegible]
[Illegible]
[Illegible]
[Illegible]

1. [Illegible]

2. [Illegible]
3. [Illegible]
4. [Illegible]
5. [Illegible]

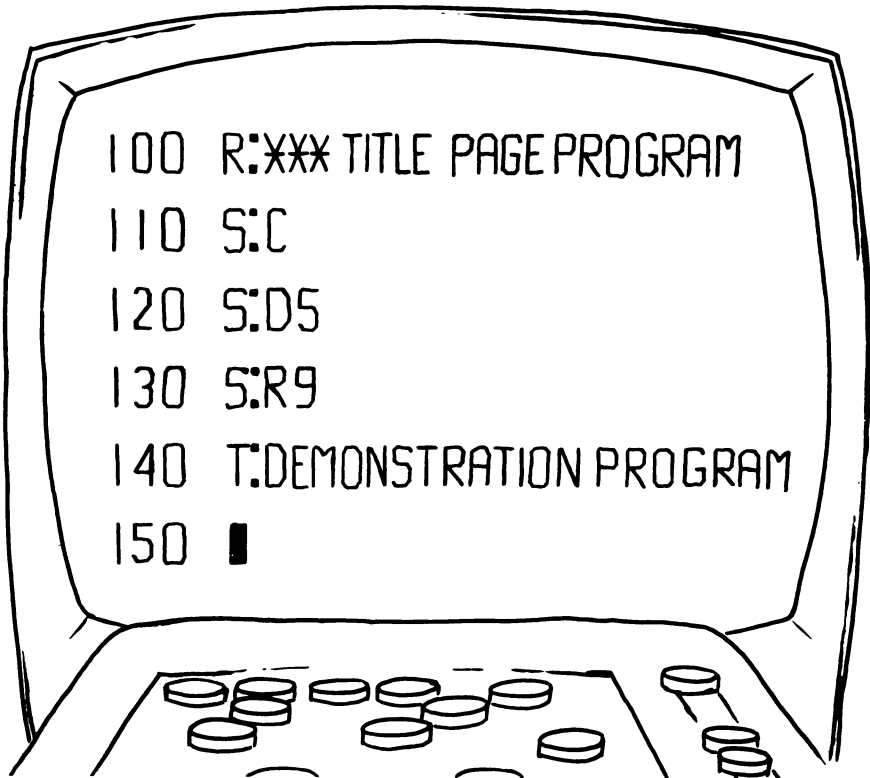
6. [Illegible]

7. [Illegible]
8. [Illegible]
9. [Illegible]
10. [Illegible]

11. [Illegible]

VI.

THE SCREEN AND SOUND COMMANDS



As you are mastering the concepts of programming, frequently you wish to dress up your programs by doing some fancy things on the screen. Vanilla Pilot has a screen command which offers a number of options for various screen functions. You can position the cursor, set the screen display mode or any of a number of other things.

The SCREEN Command -- S:

The SCREEN command offers the flexibility to do a number of things on the screen. Let's look at these in groups. The first group is the cursor control commands. The other group includes a set of miscellaneous screen aids.

Moving The Cursor

Cursor control is accomplished by the following six commands

S:C	Clear the screen and home the cursor.
S:H	Home the cursor.
S:D5	Move the cursor down 5 lines.
S:U7	Move the cursor up 7 lines.
S:R13	Move the cursor right 13 columns.
S:L9	Move the cursor left 9 columns.

The commands with the numbers are examples only. You may use any reasonable positive number for those commands. In addition, the commands can take the form

S:R#C

This will move the cursor to the right by whatever value is contained in #C.

Look at the following example

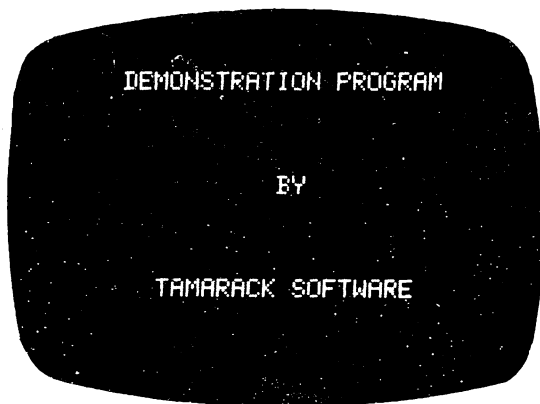
```
100 R:*** TITLE PAGE PROGRAM ***
110 S:C
120 S:D5
130 S:R9
140 T:DEMONSTRATION PROGRAM
150 S:D3
160 S:R19
```

```

170 T:BY
180 S:D3
190 S:R11
200 T:TAMARACK SOFTWARE
210 S:D7
220 T:
230 E:

```

When you run this program, the screen will clear and then look like this



This group of screen commands includes the following commands

```

S:F      Reverse the screen.
S:G      Uppercase/graphics mode.
S:N      Lowercase/upercase mode.
S:S1     Set single or double spacing.
S:V      Set the reverse flag.

```

On the VIC-20 and Commodore-64 there are two additional commands.

```

S:Bx,y   Set screen border (x), background (y)
          color on the Commodore-64.
S:Bx     Set the screen border, background
          color on the VIC-20.
S:Ox     Set character colors.

```

Let's look at each of these commands in more detail.

Screen Reversal

Try the following program

```
100 S:C
110 S:D5
120 T:HELLO
130 P:99
140 S:F
150 E:
```



What happened?

Using this program the screen will clear and the word HELLO will be typed on the 5th line. There will be a pause of about 2 seconds (the PAUSE command seen in line 130 will be discussed later). Following this the screen will reverse itself. What does a screen reverse mean? The word HELLO will now appear in dark letters on a light background. You could return the screen to its normal mode by clearing the screen OR executing another S:F command.

Screen Mode Commands

This pair of commands permits you to use the uppercase along with either graphics OR lowercase characters. The S:G command will set the graphics mode so that the computer will have available the graphics on the keyboard and uppercase letters. The S:N command will exchange 26 characters of the graphics set for the lower case letters.

This will show some of the differences between the computers in the following table:

Display Mode		
<u>Computer</u>	<u>S:G</u>	<u>S:N</u>
VIC-20	Power-on mode.	Same as using SHIFT and CBM logo keys.
Commodore-64	"	"
4000 series	"	Same as entering POKE 59468,14.
8000 series	Same as entering POKE 59468,12.	Power-on mode.
9000 series	"	"

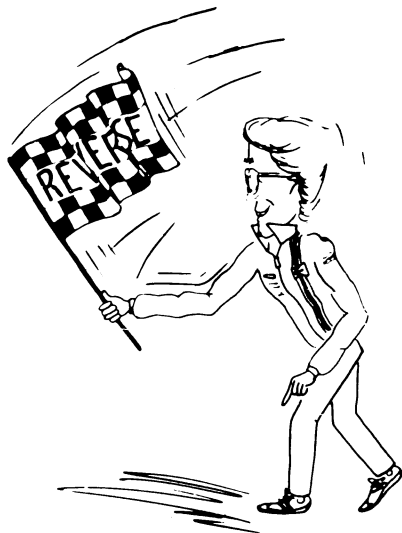
The term POWER-ON mode means the character set available when you first turn on the computer. The table describes a method of getting the alternate character set for each of the computers. Note that, for the VIC-20 and Commodore-64 computers, this command does NOT affect the use of the graphics characters shown on the LEFT hand side of the keys.

The Spacing Command

It is useful, from time to time, to have a command to permit double or single spacing of the program output. Use of the S:Sx command will permit you to switch between single and double spaced output. This is a memory conserving command. If you double space without using this command it will be necessary to insert a TYPE command every other program line. The S:S2 command will continue to output things double spaced until you issue a S:S1 command or until the program ends.

Setting the Reverse Flag

This command is different from the Screen Reverse command in that it only affects the characters typed on the next line of the screen. The reverse flag is turned off whenever the computer begins a new screen line. Using this is equivalent to pressing the **OFF/RVS** key on the PET/CBM computers and pressing the **CTRL** and **RVS ON** keys on the VIC-20 and Commodore-64 computers.



The following two SCREEN commands relate only to the Commodore-64 and VIC-20 computer

Setting the Screen Colors

Commodore-64 version

The screen colors can be set with the S:Bx,y command. The X is the background color. This can be any one of the standard color numbers between 0 and 15. The Y is the screen color and again can be any one of the standard color numbers between 0 and 15. See your computer manual for the color numbers.

VIC-20 version.

The screen background and border colors can be set with the S:Bx command. The X is any number from the table in Appendix E in your computer manual.

Setting the Character Colors

The color of the next characters you TYPE on the screen can be set by the S:Ox command. The X is any one of the color numbers from 0 to 15. This will affect the color of all subsequent characters until the next S:Ox. See your computer manual for the color numbers.

The SCREEN command includes a lot of options. Let's try a program which will make use of many of these SCREEN commands.

```
100 R:*** SPELLING PROGRAM ***
110 J:*START
120 *WAIT P:25
130 S:U1
140 R:NEXT LINE PRESS THE SHIFT AND SPACE
    BAR 25 TIMES
150 T:
160 A:$
170 E:
180 *PRESKEY S:D1
190 T:PRESS RETURN
200 A:
210 E:
220 *SETUP S:C
230 S:D3
240 T:NOW SPELL THIS WORD.
250 T:
260 E:
270 *START S:C
280 S:S2
290 T:THIS IS A PROGRAM TO TEST YOUR
300 T:SPELLING SKILLS. THE WORD
310 T:YOU ARE TO SPELL WILL APPEAR
320 T:BRIEFLY AND THEN BE ERASED.
330 T:YOU THEN MUST CORRECTLY SPELL
340 T:THE WORD.
350 U:PRESKEY
360 S:S1
370 C:Y=0
```

```
380 U:*SETUP
390 T:COMPUTER
400 U:*WAIT
410 M:COMPUTER
420 SY:V
430 TY:CORRECT!!
440 CY:Y=Y+1
450 TN:SORRY, THE WORD IS COMPUTER!
460 U:*PRESKEY
470 U:*SETUP
480 T:DISPLAY
490 U:WAIT
500 M:DISPLAY
510 SY:V
520 TY:CORRECT!!
530 CY:Y=Y+1
540 TN:SORRY, THE WORD IS DISPLAY!
550 U:*PRESKEY
560 U:*SETUP
570 T:KEYBOARD
580 U:*WAIT
590 M:KEYBOARD
600 SY:V
610 TY:CORRECT!!
620 CY:Y=Y+1
630 TN:SORRY, THE WORD IS KEYBOARD!
640 U:*PRESKEY
650 S:C
660 T:YOU GOT #Y RIGHT OUT OF 3.
670 T:KEEP STUDYING!
680 E:
```

Enter this program into your computer. Study the listing and run the program. This program illustrates almost all of the SCREEN commands, as well as many other commands. There is one new command in line 120 of this program. The P:25 command will PAUSE the program operation for about 0.5 seconds. This is discussed further in a later chapter.

The BEEP Command -- B:

The BEEP command is used to generate sounds on the various computers. It operates differently on the different computers so you should consult the appropriate section.



The Commodore - 64 Computer

The Commodore-64 computer has available three separate voices with each of the voices having separate parameter controls, except for volume. The volume control is a master control for all the voices; they cannot be controled separately.

B:parameter list

The parameters are

Attack/decay (A)	Defines how fast your note will rise to and fall from peak volume levels. Range 0 to 255.
Sustain/release (S)	How long to prolong a note at a certain volume level and release it. Range 0 to 255.
Waveform (W)	Select one of the four waveforms available. Allowed values 17,33,65,129.
Volume (M)	Set the volume of the sound synthesizer. Range is 0 to 15.
High frequency (H) Low frequency (L)	These two parameters must be specified for each note used. Both can be between 0 and 255.
Duration (D)	Set the duration of the note (use 0 to 255).

Voice (V)	Specify which voice is to be used (1,2 or 3).
High Pulse (R)	These two parameters must be specified when wave form 65 is used. Both can be between 0 and 255.
Low Pulse (P)	

Each of these parameters may be specified by preceding it with the capitalized letter shown in parenthesis above. For example

B:V1,H17,L37,D200,M5

This would turn on voice 1 with a high frequency specification of 17 and a low frequency specification of 37. It would be on for a duration of 200 units of time and have a volume of 5.

Every BEEP command MUST have the voice and frequency specified. A B: command without any parameters will turn off the synthesizer. The A, S, W, and M options must be done BEFORE actually generating any sound effects. The VOICE parameter must be specified first in any B: command. Otherwise, the computer will not know where to place the sound specifications. If the VOICE parameter is the only one listed, then the computer will repeat the last sound specified.

For more details on the actual settings to be used see Chapter 7, Appendix M and Appendix P in your Commodore-64 User's Guide. Advanced information can be found in the Programmer's Guide. By the way, don't worry about POKEs and all of the addresses mentioned in these references. Vanilla Pilot will take care of all of those details for you!

Here is an example of a simple sound effects demonstration program

```

100 R:** SHOOTING SOUND **
110 C:1=15
120 C:E=0
130 *SHOOT B:V1,W129,A15,H40,L200,M#1
140 P:2
150 C:I=I-1
160 C:$=I
170 M:#E
180 JN:*SHOOT
190 B:
200 E:

```

Try it! Then experiment with various sound effects.

The VIC - 20 Computer

This computer has a single voice with three ranges and a noise channel. Each of these ranges is controlled by entering a number between 128 and 255. This number is related to the frequency. See Chapter 5 and Appendix F of your VIC-20 manual for details on this.

The format for this command is

B:parameter list

The parameters are

Volume (V)	Set the volume of the sound synthesizer. Range is 0 to 15.
High note (H)	Set the frequency of the high voice. Allowed range is 128 to 255.
Mid note (M)	Set the frequency of the middle voice. Allowed range is 128 to 255.
Low note (L)	Set the frequency of the low voice. Allowed range is 128 to 255.
Duration (D)	Set the duration of the note. Allowed range is 0 255.

Each of these parameters may be specified by preceding it with the capitalized letter shown in parentheses above. For example

B:V12,M151,D10

This would turn on the middle voice frequency specification of 151. It would be on for a duration of 10 units of time and have a volume of 12.

Every BEEP command MUST have the volume and frequency specified. A B: command without any parameters will turn off the synthesizer. If you do not specify the duration parameters then the voice will sound until the next B: command.

Here is an example of a simple sound effect.

```
100 R:**SIREN**
110 C:I=15
120 C:E=0
130 C:F=225
140 *SIREN B:V15,H#F
150 P:30
160 B:V15,M#F
170 P:30
180 C:I=I-1
190 C:$=I
200 M:#E
210 JN:*SIREN
220 B:
230 E:
```

Try it! Then experiment with various sound effects.

The CBM and PET Computers

These computers have a single voice with a frequency control. The computers with the 12" video monitors have a chimer built in. The versions with the 9" monitor will have to have a small amplifier and speaker attached to the User Port. Your dealer should be able to help you with this.

The format for the BEEP command is

B:frequency(,duration)

The frequency is not a direct frequency, rather it is a number between 0 and 255 which the computer translates into a frequency. The frequency can be approximately calculated by the following equation

$$\frac{500000}{8(s+2)}$$

where s is the value you use for the frequency in the BEEP command.

The duration is an optional parameter. If you do not specify the duration, as a number or variable between 0 and 255, the computer will continue to make the sound UNTIL the next BEEP command. If the next BEEP command contains a duration the sound will stop.

QUIZ YOURSELF

1. Write a program to display your name in the top left corner of the screen, the middle of the screen, and the bottom right of the screen.
2. If you have a Commodore-64 or VIC-20, add color.
3. Then, using the same program, REVERSE the screen. To have the program pause for about 2 seconds, use the command P:99 before your REVERSE command. For the ambitious programmer, use the REVERSE in a subroutine and have the screen reversing several times.
4. Add sound to your program. The ambitious programmer can have the sound change everytime the screen reverses.

VII

TURTLE GRAPHICS



Chapter VII

You have made it to the fun part of Vanilla Pilot! The Turtle Graphics is one of the most enjoyable and flexible features of Vanilla Pilot. Now you can begin the exciting task of drawing figures and pictures!

The GRAPHICS Command -- G:

There are a number of capabilities with the GRAPHICS command. This command must be handled by a set of subcommands, much like the SCREEN command. These subcommands tell the computer what graphics operations to do.

Here is a list of the graphics subcommands:

CLEAR	Initialize graphics mode.
DIRECTION d	Face the turtle to d degrees.
DOWN	Place the pen on the screen.
DRAW d	Draw a line d units long.
ERASE d	Erase a line d units long.
GOTO x,y	Move the turtle to x,y.
LEFT d	Turn left d degrees.
LOCATE x,y	Find the turtle's location.
RIGHT d	Turn right d degrees.
UP	Lift the pen from the screen.

The VIC-20 and Commodore-64 have the additional commands:

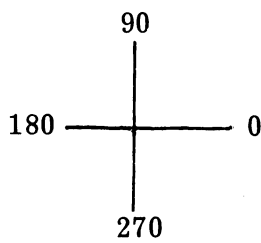
COLOR c	Set the pen to color c.
GETCOLOR x,y,c	Get the color of location x,y.
LOCATE x,y,c	Like the LOCATE above sets pen color.

Beginning concepts

Turtle graphics is a system of graphics designed by Dr. Seymour Papert and the LOGO group at the Massachusetts Institute of Technology. In the Turtle Graphics system, the computer screen becomes a playground for a tiny invisible turtle. The turtle is always at the center of an invisible circle that always moves with him. This circle is divided into 360 segments or angles, each measuring one degree, with the angle of zero degrees pointing to the right of the screen. The turtle can be facing any one of these 360 segments.

The turtle can turn left or right in any of 360 different angles. If the turtle is facing the top of the screen, he is facing 90

degrees. If he is facing the bottom of the screen, then he is facing 270 degrees. The left side of the screen is 180 degrees.



When you draw a line, the turtle moves from his current position along his current direction to a new position on the screen.

The turtle's position on the screen is determined by a system of coordinates. The top left hand corner of the screen is 0,0 while the bottom left hand corner of the screen is 0,49. The center of the screen is 39,24 (79,24 on an 80 column screen, 23,22 for the VIC-20). There is a total of 4000 points on the screen (8000 on an 80 column screen). The range is 0 to 79 (159 on the 80 column computer) horizontally and 0 to 49 vertically.

	0	1	2	3	4	5	6	7
1								
2								
3								
4								
5								
6								

The CLEAR Subcommand

This command will initialize the turtle. The format for the CLEAR subcommand is

```
G: CLEAR
```

This command does several things:

- 1) Clears the screen.
- 2) Puts the turtle at location 0,0.
- 3) Puts the pen DOWN.
- 4) Sets the turtle's direction to 0.
- 5) Sets color to 0 or black.

It is best to begin any program using Turtle Graphics with a CLEAR subcommand. That way you know exactly where the turtle is located.

The DIRECTION Subcommand

The DIRECTION subcommand will set the direction that the turtle is facing. The format is

G:DIRECTION angle in degrees

Some examples

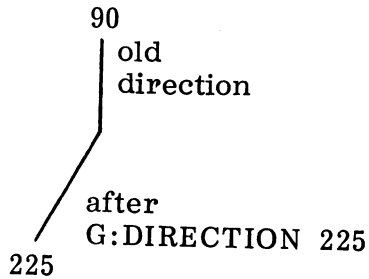
<u>COMMAND</u>		<u>DIRECTION</u>
G:DIRECTION	180	Faces the turtle left.
G:DIRECTION	315	Down and to the right.
G:DIRECTION	135	Up and to the left.
G:DIRECTION	#A	Face the direction that is specified by variable A.

This subcommand is in reference to a fixed point of reference. That fixed point is that DIRECTION 0 faces the right side of the screen and the angle increases to 359 in the counterclockwise direction. Everytime you set the DIRECTION, you will change the direction the turtle is facing to whatever angle you specify.

For example, the turtle is facing 90 degrees and you issue a

G:DIRECTION 225

The turtle will change directions from facing to the top of the screen to facing down and to the left.



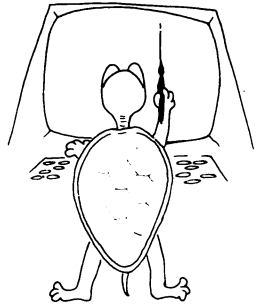
The DRAW Subcommand

The DRAW subcommand draws a line the number of units specified in the current DIRECTION. The format for the DRAW subcommand is

G:DRAW line length

Some examples are

G:DRAW 50 Draw a 50 unit line.
G:DRAW #I Draw a I unit line.



Now let's put this all together

```
100 R:*** BOX #1 ***
110 G: CLEAR
120 T: DRAW A BOX
130 G: COLOR 2
140 G: GOTO 25,25
150 G: DRAW 20
160 G: DIRECTION 270
170 G: DRAW 20
180 G: DIRECTION 180
190 G: DRAW 20
200 G: DIRECTION 90
210 G: DRAW 20
220 E:
```

This program will first clear the screen and draw a line to DIRECTION 0 and complete the box drawing a line in each of the four directions.

If you are using a 40 column or 80 column PET or CBM, do not type in line 130.

RUN the program. You should see a box on the screen.

NOTE: This program and the others in this chapter were designed for use on a 40-column screen display. If you are using a VIC-20 or 8032 computer you may need to make some adjustments to the locations and line lengths.

Also, if you are using a Commodore-64 or Vic-20, add a line for color.

The LEFT and RIGHT Subcommands

The DIRECTION subcommand is useful when turning the turtle to face another direction. Sometimes however, it is more useful to be able to turn the turtle to a direction relative to his current direction. The LEFT and RIGHT subcommands do this for you. The format is

G:LEFT 90 Turn left 90 degrees.

G:RIGHT #R Turn right R degrees.

Let's go back to the BOX program above and draw it again with the LEFT and RIGHT subcommands

```
100 R:*** BOX #2 ***
110 G:CLEAR
120 T:DRAW A BOX
130 G:GOTO 25,25
140 U:*SIDE
150 U:*SIDE
160 U:*SIDE
170 U:*SIDE
180 E:
190 *SIDE G:DRAW 20
200 G:RIGHT 90
210 E:
```

Run this program. It should draw a box just like the first BOX program. But there is one difference - This time you can draw a box at ANY angle.

Try it! Enter a new program line

```
125 G:DIRECTION 45
```

Now run the program again. This time the box should be tilted at a 45 degree angle.

If you specify any angle between 0 and 359, the box will be drawn at that angle.

The ERASE Subcommand

The ERASE subcommand is just the opposite of the DRAW subcommand. If you erase a line it will be converted to the background color. It is as if you erase a line from your paper. If you erase it properly, you will not be able to see it later. The format is

G:ERASE line length

Some examples are

G:ERASE 35 Erase a 35 unit line.
G:ERASE #K Erase a K unit line.

If you are working on the Commodore-64, the line will be set to the current background color.

To illustrate this lets modify the BOX program again

```
100 R:*** BOX #3 ***
110 G:CLEAR
120 T:DRAW A BOX
130 *START G:GOTO 25,25
140 U:*SIDE
150 U:*SIDE
160 U:*SIDE
170 U:*SIDE
180 U:*ESIDE
190 U:*ESIDE
200 U:*ESIDE
210 U:*ESIDE
220 J:*START
230 E:
240 *SIDE G:DRAW 20
250 G:RIGHT 90
260 E:
270 *ESIDE G:ERASE 20
280 G:RIGHT 90
290 E:
```

This time when you run the program you will see the box flashing on and off on the screen.

The DOWN and UP Subcommands

These subcommands direct the graphics pen to be either UP or DOWN. If the graphics pen is DOWN, then the DRAW and

ERASE subcommands will act as described. If the graphics pen is UP, then the DRAW and ERASE commands will only move the pen to a new location. There will be no apparent action on the screen. The format for these subcommands are

G:DOWN
G:UP

The COLOR subcommand

The COLOR subcommand will permit you to change the color of the pen that the turtle carries. This color number can be 0 to 15 and corresponds to the standard colors available on these computers. These colors are

VIC-20 and Commodore-64		Commodore-64 only	
0	Black	8	Orange
1	White	9	Brown
2	Red	10	Light Red
3	Cyan	11	Gray 1
4	Purple	12	Gray 2
5	Green	13	Light Green
6	Blue	14	Light Blue
7	Yellow	15	Gray 3



The default color is Black (0). The pen color remains constant until it is changed in the program.

Note that there is one peculiarity of the colors on the Commodore-64. A line is drawn using the quarter-square character. These appear as the left hand characters on the D, F, C, and V keys. There are a total of 16 possible combinations using this small square. The organization of the screen is such that, even though we may see crossing lines, there may be up to four small squares in a single character position. It is not possible to have a separate color for each of these squares. Thus, when two lines cross, the squares from BOTH may well be the color of the second line.

The LOCATE Subcommand

The LOCATE subcommand allows you to find the location of the turtle and transfer the coordinates to variables. The format is

G:LOCATE #X,#Y

This will transfer the X and Y coordinates to the variables #X and #Y. It doesn't matter which numeric variables you use, the #X and #Y variables are examples only. The LOCATE subcommand will also allow the format LOCATE #X. This will find only the X coordinate.

The VIC-20 and Commodore-64 have an additional option for the LOCATE subcommand

```
G:LOCATE #X,#Y,#C
```

If the parameter #C is specified, then the current pen color will be transferred to the numeric variable #C.

The GETCOLOR Subcommand

This subcommand will allow you to transfer the current color of a specific screen location into a variable. The format is

```
G:GETCOLOR #X,#Y,#C
```

Some formats for this subcommand are

```
G:GETCOLOR 41,25,#A    Find the color of 41,25.
G:GETCOLOR #X,18,#A    Find the color of #X,18.
G:GETCOLOR #A,#B,#C    Find the color of #A,#B.
```

where the #X and #Y are the coordinates of the screen location whose color you wish to find. The computer will look at the location on the screen and return the color in the numeric variable specified as the last variable in the list. The X and Y coordinates can be specified either as integers or as variables.

Now let's put it all together. Enter and RUN the following program you should see a beautiful flower-like design appear on the screen.

```
100 R:*** SPIRAL FLOWER ***
110 G:CLEAR
120 C:J=8
130 G:GOTO 40,25
140 G:DIRECTION 0
150 C:I=0
160 *SPLOOP U:DSPIRAL
170 G:LEFT 45
180 C:I=I+1
```

```

190 C:$=I
200 M:#J
210 JN:*SPLOOP
220 P:99
230 E:
240 *DSPIRAL C:K=0
250 *SPIRAL G:DRAW 10
260 G:RIGHT 45
270 C:K=K+1
280 C:$=K
290 M:#J
300 JN:*SPIRAL1
310 E:

```

Quiz Yourself

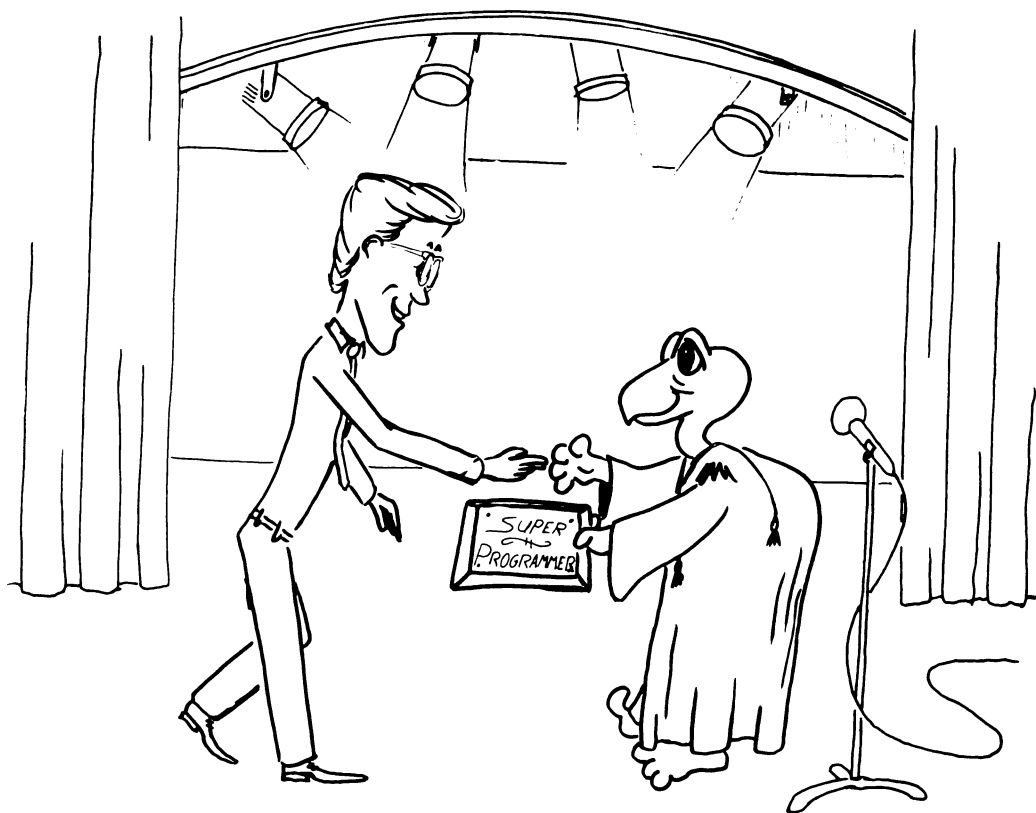
I. Match the Turtle graphics command with its function.

- | | |
|-----------------|---------------------------------|
| 1. Direction d: | - Find the turtle's location. |
| 2. Left d | - Move the turtle to x,y. |
| 3. Goto x,y | - Draw a line d units long. |
| 4. Draw d | - Turn left d degrees. |
| 5. Locate x,y | - Face the turtle to d degrees. |

II. Draw a turtle. If you have a Commodore-64 or VIC-20, color him.

VIII

SOME ADVANCED CONCEPTS



Chapter VIII

In this chapter we will cover some advanced concepts of Vanilla Pilot programming. These deal with some new commands and some new uses of familiar commands.

The PAUSE Command -- P:

The PAUSE command is a means of delaying the execution of a program. With it you have a delay range from about .02 to 2.8 seconds. The format is

P:x

The x can be either an integer or a variable with a range of 0 to 127. The time increment for each unit of change in the value of x is about .021 seconds. Thus the command

P:55

will delay the program approximately 1.15 seconds.

The NAME Field

The NAME field is a region in memory much like the ANSWER field. The ANSWER field is used whenever you type anything into the computer, and whenever you use the MATCH command. There are times, however, when you wish to enter a string, like the user's name, which will not be affected over the course of the program; this is the purpose of the NAME field.

You have only one way to enter data into the NAME field and that is via the ACCEPT command. The NAME field is designated by a question mark in the ACCEPT command (?)

A:?

With this command the keyboard input is transferred to both the NAME and ANSWER field.

When you wish to use the NAME field, you can print it in a TYPE command using the following format

T:\$?

This will TYPE the contents of the NAME field.

The following program should give you some ideas about the NAME field

```
100 R:*** NAME FIELD DEMONSTRATION ***
110 S:C
120 S:D3
130 T:HELLO THERE!
140 T:WHAT IS YOUR NAME;
150 A:?
160 T:
170 T:GLAD TO MEET YOU, $?.
180 T:BYE FOR NOW!
190 E:
```

TYPE Reserved Characters

There are three reserved characters which normally mean something to the TYPE command. These characters (#, \$, and ;) tell the TYPE command to do something special. There are times, however, when you would wish to type these characters as a part of your screen display. This can be done by preceding them with a dollar sign (\$). For example

```
100 T:YOU WON $$1.00!
```

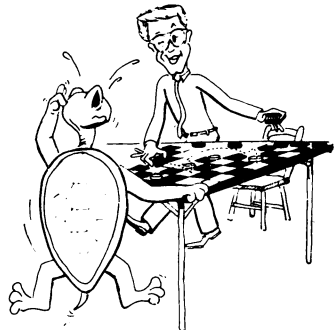
would display on the screen as

```
YOU WON $1.00!
```

Notice that the FIRST dollar sign does NOT print. The same is true for either of the other two reserved characters.

The JUMP Commands -- J:? and J:@

There are two reserved labels for the JUMP command; These are the ? and @ symbols. The format for using these are



```
J:?   Jump to the last ACCEPT.
J:@   Jump to the start of the program.
```

The first one will jump to the line containing the last ACCEPT command. Each time an ACCEPT command is executed the line number of that command is saved. Thus, the J:? is faster than a JUMP to a label.

The other, J:@, is a JUMP to the beginning of the program. It is equivalent to RUNning the program again. That is, the variables are all reset to zero and all of the other internal pointers reset to the starting values.

The WAIT Command -- W:

The WAIT command will get a single character from the keyboard. This command has several options. These are:

W:*	Check keyboard on the fly. Save any keystroke in the answer field.
W:\$	Wait for keystroke. Save character in the answer field.
W:?	Wait for keystroke. Save character in both name and answer field.
W:	Wait for keystroke. Do not save the character.

By now some of the concepts in this table should be familiar to you. Look over each of these options.

The W:* Command

This command is useful for such things as interactive animation. Here the computer checks to see if any key has been pressed. If no key has been pressed, it will go process any other necessary details, then check the keyboard again. When a key is pressed, the character corresponding to the key pressed will be saved in the ANSWER field.

The W:\$ and W:? Commands

These commands work much like the W:* command, except they will wait until a key is pressed. When the key is pressed, the value is transferred to the ANSWER field for the W:\$ command and to both the NAME and ANSWER fields for the W:? commands.

The W: Command

The W: command will wait for the key to be pressed. However, unlike the previous commands will NOT save the character corresponding to the key pressed.

Final Editor Commands

Here you have the capabilities to search the Vanilla Pilot program for a particular sequence of characters, OR to search for and then change that sequence of characters.



The FIND Command

The FIND command lets you search for a particular set of characters in the Pilot program. The format for this is

FIND /string/(,range of lines)

Some examples are:

FIND /J:*START/	Look for the command J:*START in the program.
FIND /HELLO/,500-650	Search for HELLO in lines 500 to 650.
FIND /#C/,250-	Find all occurrences of #C after line 250.

The slash (/) character is called a DELIMITER. That is, a character which surrounds or limits the set of characters you wish to find. This can be ANY character you choose, except those found in the search string. In addition, both of the delimiters must be the SAME character.

The FIND command can be useful to assist in locating specific portions of your program for study and/or modification.

The CHANGE Command

The CHANGE command is much like the FIND command in that it will allow you to find a set of characters, but includes the additional function to modify or change the character set it finds. The format for this is

CHANGE /string/string/(,range of lines)

Some examples are:

CHANGE /J:*START/J:*BEGIN/	Change every J:*START to J:*BEGIN in your program.
CHANGE /#C/#R/,500-	Change all occurrences of #C to #R after line 500.
CHANGE /HELO/HELLO/,200-400	Correct the spelling of HELLO between lines 200 and 400.

These commands will permit you to modify any program information in any portion of your program.

Quiz Yourself

I. Choose the best answer or answers for the following multiple choice questions.

1. The NAME field is used to
 - A. Store the same information that is in the ANSWER field.
 - B. Store strings which will not change during the program execution.
 - C. Store the name of the program.
 - D. Store the name of the person writing the program.
2. The JUMP command, J:?, jumps
 - A. To the last ACCEPT statement.
 - B. To the start of the program.
 - C. To the next MATCH command.
 - D. To the end of the program.
3. The CHANGE command
 - A. Changes the screen to reverse.
 - B. Changes the line numbers.
 - C. Changes the color for the VIC-20 and Commodore-64.
 - D. Finds and changes a set of numbers.

Appendix A

PILOT REFERENCE MANUAL

Appendix A, the PILOT REFERENCE MANUAL, is divided into two parts: Editor commands and Pilot interpreter commands, with the commands in each part listed in alphabetical order.

Pilot program entry is much like BASIC program entry. The Pilot interpreter makes full use of the screen editing available on the PET and CBM computers. In addition, there are a number of commands designed to make editing and debugging easier.

Some of the commands in the following list may be used with conditionals; they are enclosed in parentheses or brackets. While conditionals are not required, they add some flexibility to the operating system. Any immediate mode command or statement from BASIC is also available. All commands are to be entered using UNSHIFTED letters.

PILOT Editor Commands

AUTO nn

This command will permit automatic numbering of the program source code. To enable auto line numbering, type the command

AUTO nn

where nn is the increment between line numbers. To disable the auto line numbering, type

AUTO

with no increment.

BASIC

When you are ready to exit the PILOT interpreter and return to BASIC, type

BASIC

and the computer will transfer you back to BASIC.

CHANGE /oldstring/newstring/ (, start #-end #)

Here you can exchange all references to an old string to a new string. Unless a range of line numbers is specified, then the changes will be made through the entire source. The / above is a delimiter. This delimiter may be any character not found in either the old string or the new string.

CLOAD "Filename"

Using the command

```
CLOAD "filename"
```

will load the program "filename" from cassette number 1.

CONTINUE

By typing

```
CONTINUE
```

you can restart the execution of PILOT program, after either the RUN/STOP key has been pressed OR, after an H: statement has been executed. Editing the program, encountering a PILOT error message, or pressing RUN/STOP during a A: or W: statement will call a

```
CAN'T CONTINUE
```

error message.

CSAVE "filename"

To save a program on a cassette tape, type

```
CSAVE "filename"
```

and the program will be saved to cassette number 1.

DELETE start# - end#

The DELETE command will remove a range of the source starting with (and including) the line specified by start# and ending with (and including) the line specified by end#.

DUMP

The DUMP command will display the contents of the answer

field, name field, matchflag, and all of the numeric variables.

FIND /searchstring/ (,start#-end#)

Use this command to find and print all occurrences of a searchstring either in the entire source or within the range specified by (,start#-end#). The / above is a delimiter. This delimiter may be any character not found in the searchstring

LIST (start#-end#)

To display your PILOT program on the screen, type

LIST

This is an unformatted listing, and would appear like the short PILOT program below:

```
100 s:c
110 c:t=0
120 *rvsloop t:hello there
130 d:5
140 c:t=t+1
150 c:$=t
160 m:6
170 sy:v
180 sy:h
190 jy:rvsloop
200 e:
```

Press the space bar, and the listing will stop. Press any key, and the listing will resume.

LLIST (start#-end#)

Typing LLIST will also list your PILOT program code, but as a formatted listing. To make it easier to study your program, the various sections are put into columns. Notice when the same program LISTed above is LLISTed below.

```
100          s : c
110          c : t=0
120  *rvsloop t : hello there
130          d : 5
140          c : t=t+1
150          c : $=t
160          m : 1,3,5
170          sy: v
```

```
180 sy:h
190 jy:rvsloop
200 e :
```

LOAD “dr : filename” (,start#)

The LOAD command is exactly like the CLOAD command, but will load the PILOT source from disk device #8 instead of from cassette. If the drive number, dr, is specified, then the diskette in that drive will be searched; otherwise both drives of a dual disk will be searched. The optional parameter (,start#) will append the program being loaded from the diskette to the program in memory starting the numbering from the line number specified. If the program in memory contains line numbers with that range, the incoming program will write over them.

OFF

The OFF command turns off the TRACE function.

PLIST (start#-end#)

PLIST lists the PILOT source that is in memory to IEEE device #4. It is like the LLIST command in that the code is formatted.

Press the space bar, and the listing will stop. Press any key, and the listing will resume.

RENUMBER start#, newstart#, increment#

The RENUMBER command allows easy renumbering of the source. To use the command

```
RENUMBER
```

with no conditionals, starts renumbering at the beginning of the source giving the first line the number 100, and numbering each successive line 10 increments apart.

The conditionals allow you to start at any point in the source and renumber from there to the end. You also determine what line number the new start line will be called and the increment between lines. For example, to renumber the following program:

```
100 s:c
110 c:t=0
```



```

120 *rvsloop t:hello there
130 d:5
140 c:t=t+1
150 c:$=t
160 m:1,3,5
170 sy:v
180 sy:h
190 jy:rvsloop
200 e:

```

If the command

```
renumber 100,500,10
```

is issued, the renumbered source will be:

```

500 s:c
510 c:t=0
520 *rvsloop t:hello there
530 d:5
540 c:t=t+1
550 c:$=t
560 m:1,3,5
570 sy:v
580 sy:h
590 jy:rvsloop
600 e:

```

Line numbers are only significant when the source is being edited. All programs are renumbered when CLOAded or LOAded from the mass storage device. The interpreter ignores the line numbers when running the program.

RUN ("filename" [, start#])

The RUN command will begin executing the PILOT program in memory. If a filename is specified, then the program is LOAded from the disk and is executed immediately. If the optional parameter [,start#] is specified, then the file from disk will be appended to the program in memory BEFORE it is RUN.

SAVE "dr : filename" (, start#-end#)

The SAVE command is exactly like the CSAVE command, but it will permit saving the source onto disk drive device #3. The drive number, dr, must be specified for a proper save. If the optional parameters (,start#-end#) are specified, then only that portion of the program will be saved.

TRACE

The TRACE command prints each line of the program as it is being executed to allow you to see step by step what is happening in your program. The program lines are displayed at the top of the screen with dark characters on a light background for the CBM and PET, and with whatever the currently executed color for the VIC-20 or Commodore-64.

To start the TRACE, type

TRACE

When you type

RUN

the TRACE begins.

There are several options with the TRACE command. First, you can let the computer control the TRACE with each line executing at the speed of around one line every second. Second, you can step through the program by pressing the SHIFT key, the CONTROL key for the Commodore-64, The TRACE will then stop until you again press the SHIFT or CONTROL key. Third, to speed up the TRACE to about a fourth of normal executing speed, press the equal key for the VIC-20 or Commodore-64, and the OFF/RVS key for the CBM or PET.

UNNEW

This will permit you to recover a program, if you accidentally type NEW. This recovery can take place, only if you have not entered any program lines or LOADED a program from disk or tape.

PILOT INTERPRETER STATEMENTS

PILOT program statements all take the form:

[label] statement (conditional):(operand)

An example of this would be the line:

```
* start ty:hello there
```

On execution the label is ignored UNLESS needed by either a USE or JUMP statement. The program line would be read as follows:

Type, if the result of the last match is YES, the words 'hello there'.

Variable types and memory allocation:

- * Numeric variables are always preceded by a number sign, '#'. This is true anywhere EXCEPT in the COMPUTE statement. The COMPUTE statement ASSUMES that all variables are numbers. Numeric variables may be named using any of the 26 letters a to z. Numeric variables are limited to the range of numbers between -999 and +999.
- * The answer field is the input buffer AND workspace for the match statement. Thus, it is not directly accessible from the program.
- * The name field can be used to store any data.
- * The matchflag is used to store the result of the last match. It is unchanged until the next match is executed. On startup the match flag will match neither a yes or a no conditional. Execution of a statement will occur if no conditional is found. Otherwise the statement will be executed only if the matchflag is the SAME as the conditional.

Any statement which is not in the required format will be printed as

error - statement

This line will print the statement BEGINNING at the character where the error occurs. Other error messages are listed with the option where the message will occur.

Following is a description of the individual PILOT statements.

THE ACCEPT Command A: (operand)

When you want the user to type something during the running of a program use the ACCEPT command. Using the following operands with the ACCEPT command will also allow several types of input as follows:

- # - Accept only the numbers -999 - +999 and the minus sign (-). All other characters are ignored. Ignore the actual keystrokes.
- #x - Accept a number from the keyboard and save it in the numeric variable x.
- \$ - Accept the input of any characters on the keyboard. Save it in the answer field.
- ? - Save the input in BOTH the name and answer fields.
- <eol> - Do not save the input at all.

If you press the RUN/STOP key on the computer, the program will stop.

Error messages -

A numeric input outside the range of -999 to +999 will generate the appropriate message

VALUE < -999
VALUE > 999

and prompt for another input.

The BEEP Command B: frequency (,duration)

This routine will sound either the CB2 line or the internal beeper at frequency. If the optional parameter duration is not specified then the sound will continue until the next B: command. These values should be between 0 and 255. Any values over 255 will have 255 subtracted from them.

The Commodore-64 B: command

The Commodore-64 B: command is considerably more complex command than the CBM or PET BEEP command. Here you have eight parameters you can specify. These are

- | | |
|---|--|
| Attack/decay (A) | Defines how fast your note will rise to and fall from peak volume levels.
Range 0 to 255. |
| Sustain/release (S) | How long to prolong a note at a certain volume level and release it.
Range 0 to 255. |
| Waveform (W) | Select one of the four waveforms available.
Allowed values 17, 35, 65, 129. |
| Volume (M) | Set the volume of the sound synthesizer. Range is 0 to 15. |
| High frequency (H)
Low frequency (L) | These two parameters must be specified for each note used. Both can be between 0 and 255. |
| Duration (D) | Set the duration of the note.
(use 0 to 255) |
| Voice (V) | Specify which voice is to be used (1, 2 or 3). |

Each of these parameters may be specified by preceding it with the capitalized letter shown in parenthesis above. For example

B: V1, H17, L37, D200, M5

This would turn on voice 1 with a high frequency specification of 17 and a low frequency specification of 37. It would be on for a duration of 200 units of time and have a volume of 5.

Every BEEP command MUST have the voice and frequency specified. A B: command without any parameters will turn off the synthesizer. The A, S, W, and M options must be done BEFORE actually generating any sound effects. The VOICE

parameter must be specified first in any B: command, Otherwise, the computer will not know where to place the sound specifications. If the VOICE parameter is the only one listed, then the computer will repeat the last sound specified.

The VIC-20 B: Command

This VIC-20 computer has a single voice with three ranges and a noise channel. Each of these ranges is controlled by entering a number between 128 and 255. This number is related to the frequency. See Chapter 5 and Appendix F of your VIC-20 manual for details on this.

The format for this command is

B:parameter list

The parameters are

Volume (V)	Set the volume of the sound synthesizer. Range is 0 to 15.
High note (H)	Set the frequency of the high voice. Allowed range is 128 to 255.
Mid note (M)	Set the frequency of the middle voice. Allowed range is 128 to 255.
Low note (L)	Set the frequency of the low voice. Allowed range is 128 to 255.
Duration (D)	Set the duration of the note. Allowed range is 0 to 255.

Each of these parameters may be specified by preceding it with the capitalized letter shown in parenthesis above. For example

B:V12,M151,D10

This would turn on the middle voice frequency specification of 151. It would be on for a duration of 10 units of time and have a volume of 12.

Every BEEP command MUST have the volume and frequency specified. A B: command without any parameters will turn off the synthesizer. If you do not specify the duration parameter, then the voice will sound until the next B: command.

The COMPUTE Command C: equation

The COMPUTE command can accept a variety of operations. Acceptable formats for the equations are:

a=b+c
a=b+50
a=b-c
a=b-50
a=-b+75
a=-b-75

The first six equation types are samples of the possible operations. Multiple additions and subtractions are possible. Any combination of integers and numeric variables can be used. No multiplication or division is available.

\$=a

The \$ is required before using a match with a numeric variable. Transfer the number to the answer field.

Error message -

If the result of the calculation is outside the range of -999 to +999, the interpreter will print the error message:

error - c:equation
overflow in calculation

The END Command E:

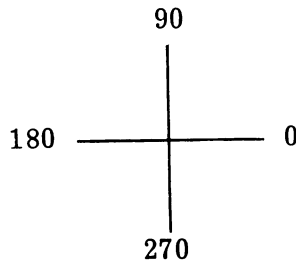
The END statement will either end the program OR return to the main program from the last open USE (or subroutine) statement.

The Deferred Mode TRACE Command!:

The deferred mode TRACE statement will turn the TRACE function on, if it is off, or off, if it is on. Use the deferred mode TRACE to trace through a section of the Pilot program source.

The GRAPHICS Command G: subcommand

This command allows entry to the graphics processor. The processor uses the quarter square graphics for a resolution of 50 vertical by 80 horizontal points on the Commodore-64 and 40 column screens, 44 by 46 on the VIC-20, and 50 by 160 on a 8032. The coordinate system has locations 0,0 at the upper left corner of the screen. The directions are as follows:



When the interpreter is initialized, or turned on, the direction is set to 0.

The command may be one of the following.

- CLEAR - Clear the screen and set the pen coordinates to 0,0 (upper left screen corner).
- COLOR x - On the VIC-20 and Commodore-64 will set the turtle pen color to color number x.
- DIRECTION - Set the turtle direction between 0 and 359 degrees.
- DOWN - Lower the drawing pen to enable a line to be drawn.
- DRAW x - Draw a line x units long from the current turtle position in the current DIRECTION.
- ERASE x - Erase a line x units long from the current turtle position in the current DIRECTION.
- GOTO x,y - Move the turtle coordinates to x,y.
- LEFT x - Adjust the current DIRECTION x degrees to the turtle's left (or counter-clockwise direction).

- LOCATE** - Locate the x and (optionally) the y coordinates of the turtle and transfer them to the PILOT variables x and y. It is not required that you use x and y as variable names.
- RIGHT x** - Adjust the current DIRECTION x degrees to the turtle's right (or clockwise direction).
- UP** - Raise the drawing pen. When this command is in effect, the DRAW command will not draw a line and the ERASE command will not erase a line. The only effect with the DRAW and ERASE commands will be that the position of the turtle will be moved as if the lines had been drawn or erased.

On PILOT startup or initialization the interpreter will set the pen coordinates to 0,0 and put the pen DOWN and set the COLOR to 0.

The HALT Command H:

The HALT command will stop execution of the PILOT program so you can examine the program steps OR variables. Type CONTINUE to restart the program. If you edit a program line, the CONTINUE command will respond with a

CAN'T CONTINUE

error message. You will then need to restart the program by typing RUN.

The JUMP Command J: label

The JUMP command jumps to the appropriate label and begins execution of the program there. There are two labels reserved for the JUMP Command:

J:@ - Equivalent to typing RUN from the keyboard. It will restart the program AND clear all variables.

J:? - Jump to the last accept statement.

Any other label will be assumed to be a part of the users program. The search for the label will proceed from the beginning of the program and continue until all of the characters in the operand field are matched. This condition can be met even if the destination label is longer than the one in the JUMP statement. Thus, the JUMP statement

j:test

will jump to

*test loop t:go jump in the lake

if the label * testloop is the first match found. The computer will match only the first four characters, even if a later label is

*test t:Press any key to continue

and is the one you are wanting to match.

Error messages -

If the destination label or the label in the JUMP statement is longer than 8 characters, then the error message

label - label too long

will be printed. If the label is not found, then the message

label - label not found?

will be printed.

The MATCH Command

M: item 1, item 2,.. . ., itemn

The MATCH command will match the contents of the answer field with item1. If no match is found, then match with item2, etc. If no match is found, then set the matchflag to n. If a match is found, then set the matchflag to y.

The items to be matched can also be variables. If the first character following the colon or following each comma is a number sign '#', then the answer field will be matched with the number.

The literal matches use a sliding window match. That is:

test will match test
OR that was a thorough testing program
OR uncontestable

The RANDOM NUMBER Command N:X

The RANDOM NUMBER Command generates a random number and transfers that number to the variable x. The number generated will be between 0 and 99.

The PAUSE Command P:nn

The PAUSE Command lets you have a delay at some point in the program for a period of time. The value of nn can be anywhere from 1 to 127. Each increment (from 1 to 2, 2 to 3, etc.) will delay for about .02 seconds. Thus, a delay of 127 will pause for about 2.7 seconds.

The REMARK Command R:remarks

The REMARK statement is a convenient way to add comments to your program to help you or others understand what is happening. The interpreter will skip this line when executing a program.

The SCREEN Command S: (options)

The SCREEN command will send various formatting commands to the screen. These commands are:

- | | |
|-----|--|
| c | - Clear the screen. |
| dnn | - Cursor down nn times (Note 1). |
| f | - Reverse the screen. |
| g | - Switch to uppercase/graphics. |
| h | - Home the cursor. |
| lnn | - Cursor left nn times (Note 1). |
| n | - Switch to lowercase/uppercase. |
| rnn | - Cursor right nn times (Note 1). |
| snn | - Set single or double spacing (Note 2). |
| unn | - Cursor up nn times (Note 1). |
| v | - Set reverse flag. |

Option unique to the Commodore-64

b x,y - set screen background, border colors (Note 3).

Option unique to the VIC-20

b x - set screen background & border colors (Note 3).

Option for both the Commodore-64 and VIC-20

onn - set character colors (Note 3).

NOTE 1 - The value nn can either be an integer or a variable, but it must not be more than 99.

NOTE 2 - The value nn must be either a 1 or 2. Any other values are ignored. Integer or variable values can be used.

NOTE 3 - The color numbers must be between 0 and 15 inclusive on the Commodore 64 and 0 to 7 on the VIC-20.

The TYPE Command T: (output)

The TYPE Command will print on the screen anything following the colon. If you include any of the following, other items can be printed as well:

#x - Print the contents of numeric variable x.

\$? - Print the contents of the name field.

If you wish to print a # or a \$, then precede it with a \$. The line

```
100 t:you won $$100!
```

would appear on the screen as

```
you won $100!
```

If the last character of the line is a semicolon (;), the computer will not jump to the next screen line. Rather the next T: statement will begin printing at the next screen location. If the semicolon appears in the text of the T: statement, then it will be printed.

The USE Command U: label

The USE Command will transfer control of the program to the subroutine which has the appropriate label. Subroutines can be nested to a depth of seven.

When an E:command is encountered, control is returned to the statement immediately following the USE statement.

Error messages -

In addition to error messages listed under the JUMP statement, the message

label - use depth exceeded

will be printed, when you pass the maximum of seven nested subroutines.

The WAIT Command W:

The WAIT Command needs a keystroke from the keyboard before proceeding to the next command. The options are:

- * - Save the keystroke in the answer field. Don't wait for the keystroke.
- \$ - Save the result in the answer field. Wait for the key to be pressed.
- ? - Save the keystroke in both the answer fields AND the name field. Wait for the key to be pressed.
- <eol> - Don't save the keystroke value. Wait for the key to be pressed.

Item 10 Bush 101 101

advertising and promotion of the product
and the company's name and logo
and the company's name and logo

advertising and promotion of the product
and the company's name and logo
and the company's name and logo

advertising and promotion of the product
and the company's name and logo
and the company's name and logo

advertising and promotion of the product
and the company's name and logo
and the company's name and logo

advertising and promotion of the product
and the company's name and logo
and the company's name and logo

advertising and promotion of the product
and the company's name and logo
and the company's name and logo

advertising and promotion of the product
and the company's name and logo
and the company's name and logo

advertising and promotion of the product
and the company's name and logo
and the company's name and logo

advertising and promotion of the product
and the company's name and logo
and the company's name and logo

advertising and promotion of the product
and the company's name and logo
and the company's name and logo

advertising and promotion of the product
and the company's name and logo
and the company's name and logo

APPENDIX B

PREPARING A DISKETTE FOR USE

For the Commodore-64 and VIC-20 place a blank diskette in the drive. Then type

```
OPEN 1,8,15,"n0:diskname,10":CLOSE1
```

This command will format the diskette and give it whatever name you enter in place of the "diskname" above. The '10' is the disk identification number. You can use any TWO digit number here. After about two to three minutes, the disk drive will stop running. The diskette will now be ready to store your Vanilla Pilot programs.

The same commands can be used with the 4032 and 8032 computers. However, you may find that an alternate command somewhat easier to use. This command is

```
HEADER "diskname",i10
```

This tells the computer to format the diskette and give it whatever name you enter in place of the "diskname". The disk identification number follows the letter i.

When you press the RETURN key, the computer will respond with a question

```
ARE YOU SURE ?
```

To format the diskette you need to press the Y key and then RETURN .

APPENDIX C

PILOT ERROR MESSAGES

There are a number of error messages that you might see with the PILOT interpreter. Most of these have been designed to be virtually completely self explanatory. What follows is a list of the error messages from PILOT and their possible causes.

EDITOR ERROR MESSAGES

- | | |
|----------------------|--|
| FILE NOT FOUND | - The requested file was not on the diskette. Try a different name OR a different diskette. |
| FILE ERROR | - There was an error when attempting to READ a file from the disk. Usually this means a disk problem. First try reseating the diskette. Then clean the disk heads. Finally, try LOADING from the backup copy of your diskette. |
| FILE WRITE ERROR | - There was an error on the diskette while attempting to SAVE a file on the disk. Try SAVEing on a different diskette, or try some of the other ideas in the FILE ERROR message. |
| CAN'T CONTINUE | - You have attempted to continue after LOADING a new program, editing the program in memory, or pressing the RUN/STOP key while executing an ACCEPT or WAIT statement. |
| NOT PILOT
COMMAND | - You have attempted to use one of the BASIC commands which PILOT does not understand. |

ZERO INCREMENT

- On the CBM or PET (the Commodore 64 and the VIC-20 give a SYNTAX ERROR), you have attempted a RENUMBER command with an increment of ZERO. This would give all the program lines the same number and could confuse the computer and possibly the programmer.

SYNTAX ERROR

- Usually this means that you have misspelled a command OR have entered or omitted a required parameter. When using the Commodore 64 or the VIC-20, this can also mean the same as the ZERO INCREMENT error message.

INTERPRETER ERROR MESSAGES

Most errors in the interpreter and the Turtle Graphics portions of Vanilla Pilot will simply look like this

ERROR - program line

The program line portion of the error message will show the line STARTING from the point where the error will occur. This is sometimes given in conjunction with the error messages below.

- | | |
|-------------------------------|--|
| label - LABEL NOT
FOUND | - A J:label or U:label contains a label which does not exist in the program. The label displayed will be the one the computer couldn't find. |
| label - LABEL TOO
LONG | - You have entered a label longer than 8 characters. The label displayed will be the first eight characters of the label that is too long. |
| label - USE DEPTH
EXCEEDED | - The program has eight or more nested subroutines. The displayed label is the eighth nested subroutine. |

- | | |
|----------------------------|---|
| VALUE < 0 | - A required parameter is less than zero. |
| VALUE < -999 | - There was an attempt to enter a number less than -999 in an ACCEPT command. |
| VALUE > 999 | - There was an attempt to enter a number greater than 999 in an ACCEPT command. |
| OVERFLOW IN
CALCULATION | - The result of the COMPUTE command is outside the range of -999 to 999. |

TURTLE GRAPHICS ERROR MESSAGES

- | | |
|---------------------------|--|
| TURTLE OFF THE
SCREEN | - The DRAW, ERASE or GOTO subcommand has sent the turtle off the graphic screen. |
| TURTLE DIRECTION
> 359 | - You have entered a DIRECTION command with the direction specified as greater than 359 degrees. |
| ZERO LINE LENGTH | - A DRAW or ERASE command was entered with a value of zero for the line length. |
| VALUE MISSING | - You have entered one of the graphics subcommands without a parameter. |

APPENDIX D

CONTROLLING THE JOYSTICK ON THE COMMODORE - 64 AND VIC - 20

The joystick and fire button ports on the Commodore-64 are read by the Y: and I: commands. These commands are covered in this appendix.

The JOYSTICK Command - Y:

The JOYSTICK command will permit you to read the direction registers of the joystick and convert them to Pilot variables. The format for this command is

Y:port,variable1,variable2

The port is either 1 or 2 depending on which joystick you wish to read. The variables will contain the current status of the joystick. The following table will give the values transferred to variable1 and variable2

Direction	variable1	variable2
center	0	0
up	0	-1
up & right	1	-1
right	1	0
down & right	1	1
down	0	1
down & left	-1	1
left	-1	0
up & left	-1	-1

A typical example of this command would be

Y:1,#X,#Y

This means to transfer the relative directions for joystick #1, from the table above, to the variables #X and #Y. Look at this simple example

```

100 R:SIMPLE JOYSTICK DEMO
110 S:C
120 C:Y=12
130 C:X=20
140 *START S:H
150 S:D#Y
160 S:R#X
170 T:*
180 Y:1,#A,#B
190 C:X=X+A
200 C:Y=Y+B
210 J:*START
220 E:

```

This program would place an asterisk in the center of the screen. As you move the handle of joystick #1 you will see the asterisk move about the screen leaving a trail of asterisks behind it.

VIC-20 Note

The VIC-20 version of the joystick command is identical except for the port number. As there is only one joystick connection, you need not specify joystick #1 or #2.

The FIRE BUTTON Command - I:

This command will permit you to check the status of the fire button on either joystick. The format is

I:port,variable

The port is either a 1 or a 2 depending on which fire button you wish to read. The variable is any one of the 26 numeric variables from Vanilla Pilot.

A typical example of this is:

I:2,#F

This line will read the fire button on joystick #2. The variable #F will contain a 0 if the button was not pressed. If the button was pressed, then #F will contain a 1.

Let's expand the JOYSTICK example to provide an example of how to use the FIRE BUTTON command.

```

100 R:SIMPLE JOYSTICK DEMO
110 S:C
115 C:C=0
120 C:Y=12
130 C:X=20
140 *START S:H
150 S:D#Y
160 S:R#X
165 S:O#C
170 T:*
180 Y:1,#A,#B
190 C:X=X+A
200 C:Y=Y+B
202 I:1,#F
205 C:C=C+F
210 J:*START
220 E:

```

Now when you RUN the program, you will see the asterisk moving about the screen as directed by the joystick, just as before. Now, however, pressing the button on the joystick will change the color of the asterisk. Each time you press the button the color will increment by 1.

VIC-20 Note

The VIC-20 version of the fire button command is identical except for the port number. As there is only one joystick connection you need not specify fire button #1 or #2.

QUESTIONNAIRE

We would like your opinion on this program, so that in future editions it can be improved.

1. Did you find any errors in the book? ____ Yes
____ No. If yes, please explain

2. Did you find any errors in the program? ____ Yes
____ No.

If yes, please explain _____

3. Was the book written on a level that you could understand? ____ Yes ____ No. If no, what concepts did you have difficulty understanding?

How could these concepts be explained better?

4. After working with the examples in the book, do you feel confident that you can write programs in Vanilla Pilot? ____ Yes ____ No.

5. Do you think that any area of the manual should be expanded? ____ Yes ____ No. If yes, which area(s)

6. Other comments _____

(Optional)

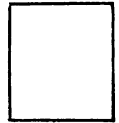
Name _____

Address _____

City _____ State _____ Zip Code _____

Phone (_____) _____

-----Fold Here-----



Tamarack Software

Box 247

Darby, MT. 59829

Welcome to VANILLA PILOT.

VANILLA PILOT is a simple easy-to-learn programming language. The commands have meaningful names, making them easy to learn and remember. Vanilla Pilot has been developed for teachers to write programs for their students and for teachers to teach their students. It also is ideal for someone who wants to learn programming at home.

VANILLA PILOT is a versatile language. It contains a wide range of program commands with many useful editing statements. Using the Turtle Graphics commands are both easy and fun. On the Commodore-64 and VIC-20, there is the added dimension of color. Also included are sound and joystick commands.

This manual is planned for both the novice and the more advance programmer. The first eight chapters are a hands-on tutorial for the beginning programmer and could easily be used as a teaching guide. Following the explanation of each program feature are examples for the user to try. There are numerous illustrations and computer video screens to aid in understanding what should be happening.

If you already know Pilot, there is a section listing the various commands with an explanation of how each one works.

Tamarack Software
Darby, MT. 59829