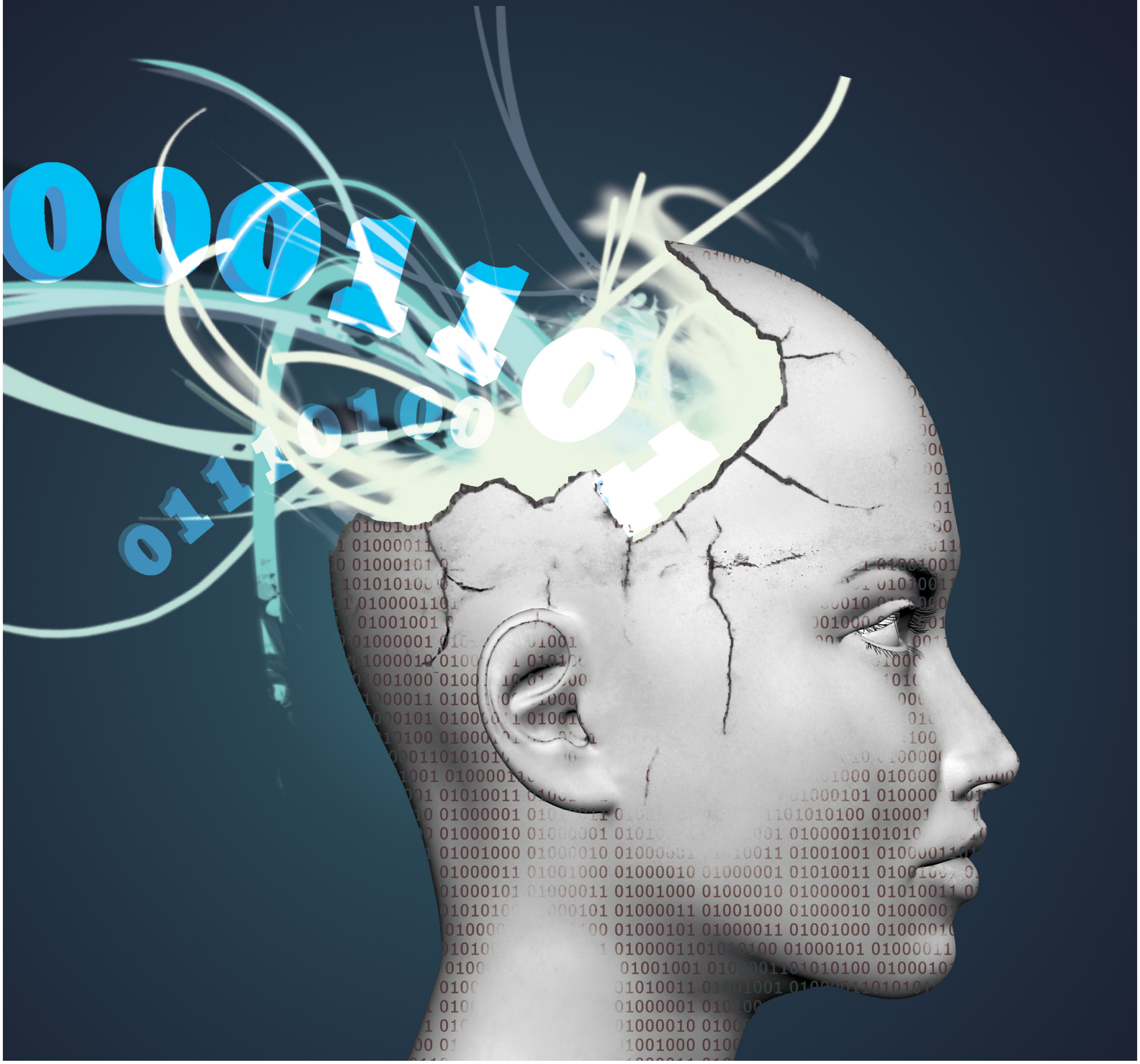
techBASIC™ 3.3

Quick Start Guide for iPad



A product of

Byte Works®, Inc.

http://www.byteworks.us

Credits

techBASIC Programming

Mike Westerfield

techBASIC Art

Karen Bennett

Documentation

Mike Westerfield

What is techBASIC?

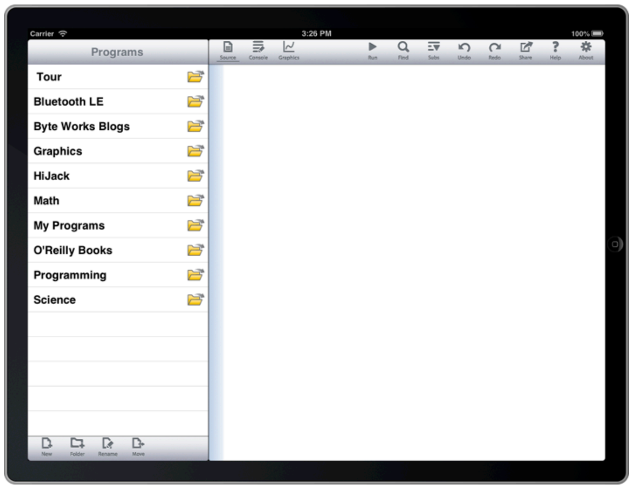
techBASIC and techBASIC Sampler are all about writing programs on the iPad, especially programs that manipulate numeric information—collecting it from sensors like the iPad accelerometer, processing it with a powerful implementation of the BASIC computer language that includes advanced matrix manipulations, and displaying the results in plots that you can twist and zoom to explore the nuances of the data. You can, for example, collect information about G forces as you hop up and down or twirl in a pirouette, then plot the time-based data—and in a moment, you'll do just that!

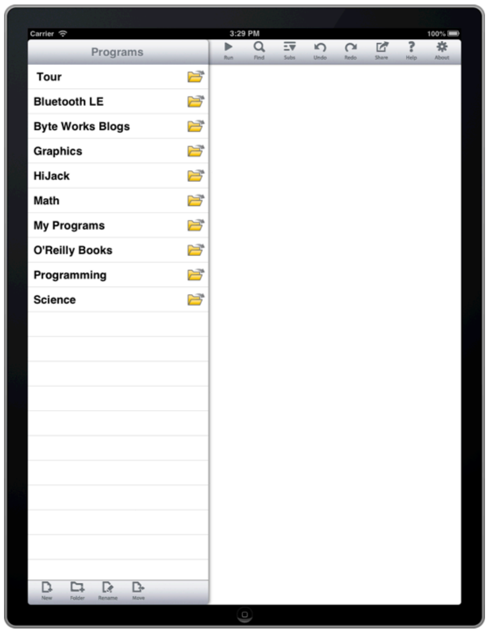
This guide covers both techBASIC and techBASIC Sampler. They are, for the most part, the same program. techBASIC is a paid program that gives you all of the power of a technical computing environment right away, while techBASIC Sampler is a free demo version that disables editing existing programs or creating new ones. You can upgrade techBASIC Sampler to have all of the capabilities of techBASIC at any time through an in-app purchase. For the rest of this guide, we'll refer to techBASIC, but except for editing, everything applies equally well to techBASIC Sampler.

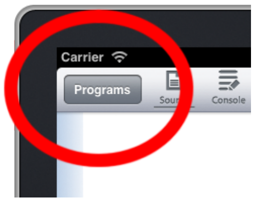
This quick start guide is designed to get you up and going on techBASIC right away. It takes just a few minutes to read, but shows you some of the capabilities you can use right away, and where to go for more details as you need them. We hope you'll read this guide, then toss it aside as you move on to explore the program.

Manipulating Information

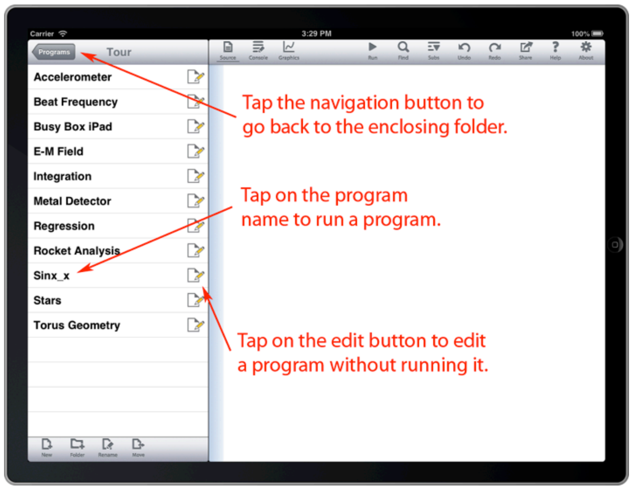
When you start techBASIC, you'll see a display like one of these, depending on which way you hold your iPad:







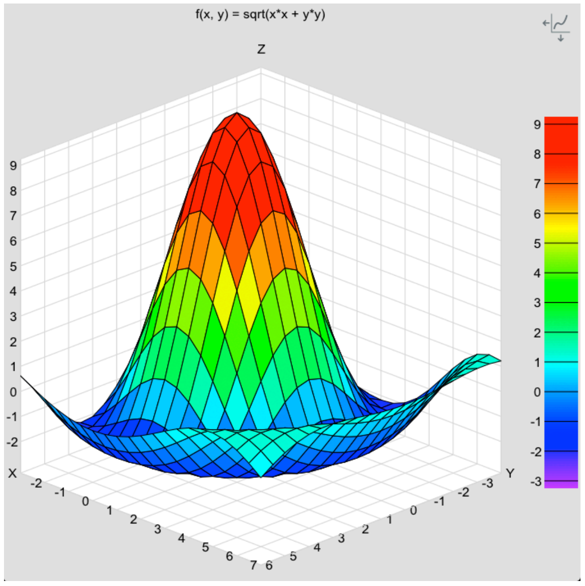
A list of folders programs is on the left side of the screen for landscape view, and available with a touch of the Programs button in portrait view. You can dismiss the list in portrait view by tapping outside of the list. At first, only folders are visible, as indicated by the folder icon to the right of the name. Tap on either the folder icon or the name to open the folder, revealing the programs and folders imbedded in the folder. For now, tap the Tour folder.


Programs
Tour
Accelerometer
Beat Frequency
Busy Box iPad
E-M Field
Integration
Metal Detector
Regression
Rocket Analysis
Sinx_x
Stars
Torus Geometry
Tap the navigation button to go back to the enclosing folder.
Tap on the program name to run a program.
Tap on the edit button to edit a program without running it.

Navigating through the folders is pretty easy. You've just seen how tapping the name of a folder opens it. Tapping the back button at the top left of the screen backs up a level.

Programs have an edit document button to the right of their name instead of a folder button. Tap the edit document button to see the source for a program. Tap the name of a program to run it. Try running Sinx_x.

Visualizing 3D

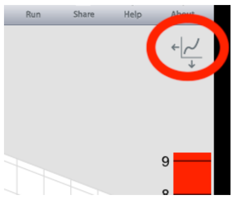


When Sinx_x runs, you'll see this rainbow colored plot of 10*sin(x)/x. This is just one of several axis styles and coordinate systems supported by techBASIC.

Move it around a bit to see various parts of the function. Dragging up and down moves the function along the Z axis. Dragging parallel to the X or Y axis drags the function along one of those axis. As you move, the coordinates along the edge of the bounding box update, as does the legend to the right of the plot. If you get lost, rerun the program—it doesn't take long.

Use two fingers to pinch or expand the plot. Zoom in a lot—techBASIC recomputes the function as you move or zoom, showing as much detail as you like with no loss of accuracy or increase in pixilation.

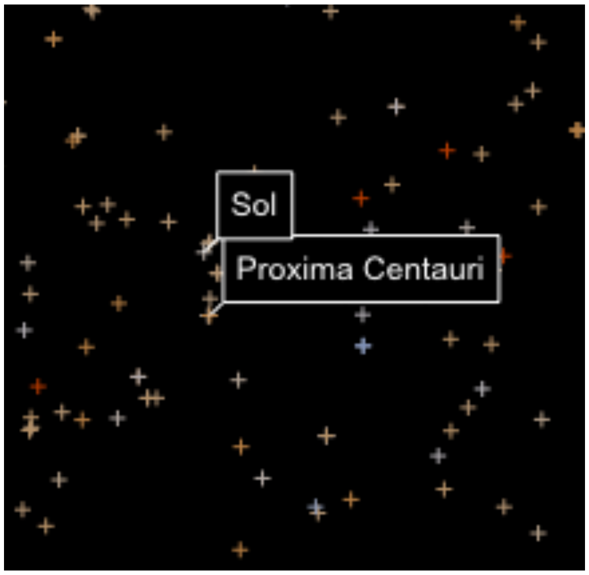
Now take it for a spin—literally! Using two fingers held apart like you are going to start a pinch, twist instead. The function will turn about an axis projecting out of the screen. To twist it about an axis in the screen, use two fingers and drag as if you were trying to turn a sphere with the plot inside. As you can see, it's easy to manipulate functions to see them from any angle or level of detail.



So far, you're moving the plot within a fixed axis. Sometimes part of the plot is off screen, or you might want to zoom in by enlarging the axis itself. Tap the pan/zoom button at the top right of the plot, and swipe and pinch gestures apply to the axis itself, not the function. Tap it again to restore the original behavior.

Tap on the plot. techBASIC will show you the point on the mesh closest to where you tapped, giving the X, Y and Z coordinates. The callout will move around with the plot as you move the plot itself. Tap on the callout to dismiss it. You can add as many as you like.

Starry Night



But you're not limited to functions. Run the Stars sample, which loads in all of the stars within 10 parsecs of our sun. Use the two-finger drag you learned about with the Sinx_x sample. Suddenly the static, two-dimensional display of stars jumps to life, allowing you to see the special relationship between all of those stars.

But which star is which? Tap on one, and a callout pops up showing the name of the star. The one at the origin is Sol, our own star.

Behind the Curtain

That's a lot of sophisticated graphing, so it must take a lot or programming, right? Let's take a look. Tap the edit program button for Sinx_x, or if it was the last program executed, just type the Source button at the left of the button bar. Here's the program that generated the plot:

```
   ! Display the graphics view.
   system.showGraphics

   ! Set up the plot, label it, and display the function
   ! with false color.
   DIM p AS Plot
   p = graphics.newPlot
   p.setTitle("f(x, y) = 10*sin(r)/r; r = sqrt(x*x + y*y)")
   p.setGridColor(0.85, 0.85, 0.85)
   p.setsurfacestyle(3)
   p.setAxisStyle(5)

   ! Add the function.
   DIM func AS PlotFunction
   func = p.newFunction(FUNCTION f)

   ! Adjust the function so the portion under the X-Y
   ! plane is visible, and push it slightly off the Z
   ! axis so the beginning of the descending curve
   ! can be seen.
   p.setTranslation3D(-4, -3, -2)
   p.setScale3D(1, 1, .8)

   ! Show the about box. Comment out the next line to prevent the About alert
   ! from showing up when the program starts.
   showAbout
   END


   ! Shows the About alert when the program starts.
   !
   SUB showAbout
   about$ = "Here's a simple way to visualize functions in 3D. Swipe with one
finger, or use two to rotate. Pinches enlarge or shrink the plot. Tap the
```

```
button at the top right of the display to switch from changing the plot to
changing the axis."

   about$ = about$ & CHR(10) & CHR(10) & "The function that is plotted is
right at the end of the program. Change it and you change the plot to
whatever function you want to visualize."

   about$ = about$ & CHR(10) & CHR(10) & "To disable this alert, comment out
the"
   about$ = about$ & " line ""showAbout""."

   i = Graphics.showAlert("About This Sample", about$)
   END SUB


   ! This is the function the program plots.
   !
   FUNCTION f(x, y)
   d = SQR(x*x + y*y)
   IF d = 0 THEN
     f = 10
   ELSE
     f = 10*SIN(d)/d
   END if
   END FUNCTION
```

Most of the code consists of comments, and a lot of what is left is actually just selecting various styles for the axis, picking colors, and so forth. The entire `showAbout` subroutine just pops up the informative alert you saw when the program started. All we really need to see the function is this:

```
   DIM p AS Plot
   p = graphics.newPlot
   DIM func AS PlotFunction
   func = p.newFunction(FUNCTION f)

   FUNCTION f(x, y)
   d = SQR(x*x + y*y)
   IF d = 0 THEN
     f = 10
   ELSE
     f = 10*SIN(d)/d
   END if
   END FUNCTION
```

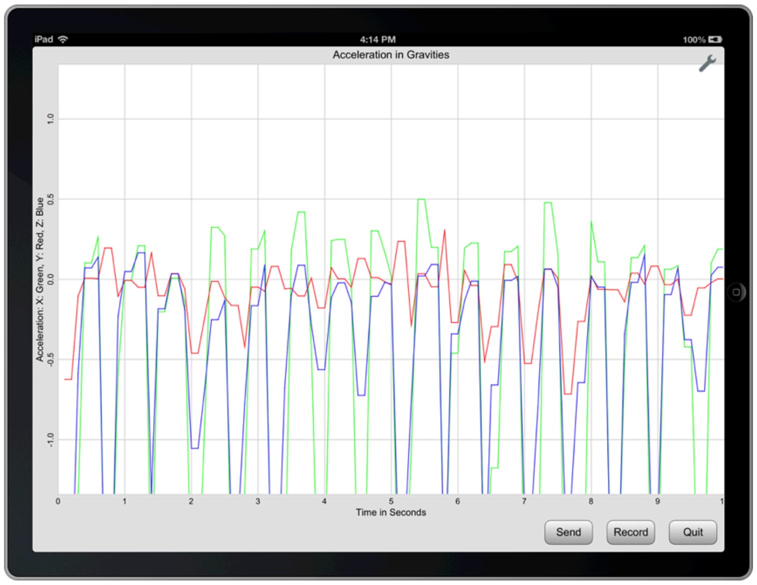
The first two lines create the plot itself—the area that appeared on the graphics screen. The next two add a function to the plot. In this case, the function has two variables, so techBASIC knows it needs to be plotted in three dimensions. The function itself can be as simple or as complicated as you like. techBASIC calls the function for each point on the mesh to plot the function you see. To plot another function, just replace what you see in the function itself with your own and rerun the program.

Did you notice how easy it is to change this program to visualize any function you like? Change the function at the end of the program, and the program will plot any 3D function. Remove the Y value from the function and it will plot 2D functions.

The rest of the program does some cool things, but isn't really needed. Later we will explore the help system and the reference manual, either of which will tell you about the other commands.

Data From Your Accelerometer

Your iPad has a built-in accelerometer that is usually used for mundane things like deciding which way the iPad is turned. It's also a great sensor for all sorts of experiments. We'll use a simple program to see how this works.



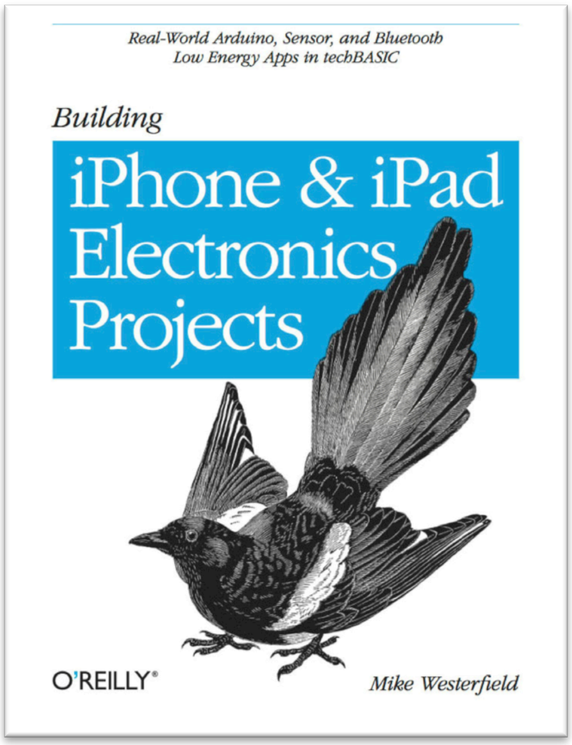
Accelerometer measures the G force along three axis, plotting the last 10 seconds of data as it runs. Run the program and jump up and down for a while. Go ahead—no one's looking. The plot to the right shows my results.

There are three buttons along the bottom you might want to try. The middle one starts a recording of the data, and stops it when you tap the button a second time. Once you have some data recorded, pressing Send bundles it up into a nice package and puts it in an email message so you can send it elsewhere for further analysis. Since the program takes over the entire screen, there is also a Quit button to stop the program.

Looking inside the program, there is one line worth a more detailed look. Right near the front you see the line

```
a = Sensors.accel
```

There's a lot hiding in that simple line. It turns on the accelerometer, reads the acceleration along all three axis plus a time stamp, and returns the values in an array. You could add a PRINT statement, like this, and have a complete, working program.


Real-World Arduino, Sensor, and Bluetooth Low Energy Apps in techBASIC
Building
iPhone & iPad Electronics Projects
O'REILLY
Mike Westerfield

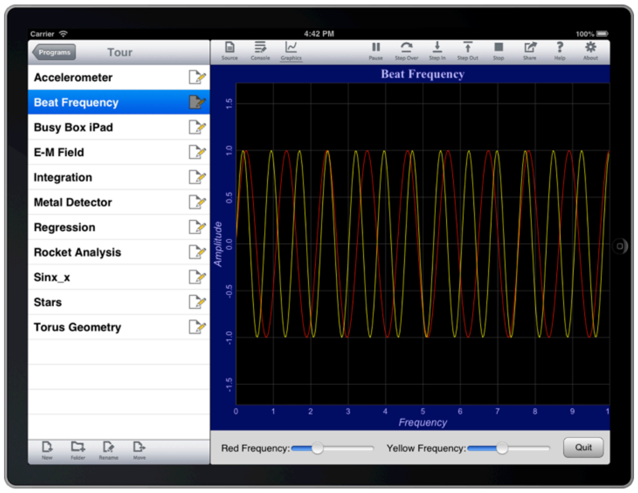
```
a = Sensors.accel
PRINT a
```

This program not only reads the acceleration, it prints the values to the console. Accessing sensors in techBASIC is usually trivial. The rest of the program is just setting up the GUI, implementing data recording, allowing you to send an email, and so forth.

Of course, it's fair to ask where you would learn what the rest of the program does. One way is to look through the source, reading the comments. All of the samples are thoroughly commented to help you learn programming in techBASIC quickly. For this particular sample, another way is to get the book *Building iPhone & iPad Electronics Projects*, currently available as a prerelease electronic book from O'Reilly Media and due out in print form in September 2013. This program is a sample in Chapter 1.

Completing the Tour

The samples in the Tour folder are designed to give you a good overview of the capabilities of techBASIC. You've already seen how easy it is to create and manipulate plots when we looked at Sinx_x. You saw how complex information, like the orientation of the stars close to the Sun, can be manipulated. You also saw how easy it is to access sensors from techBASIC. Here are a few more representative samples from the Tour folder.

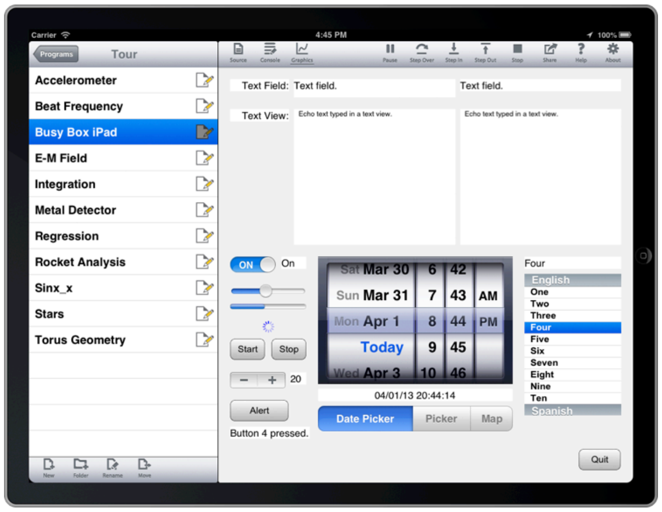


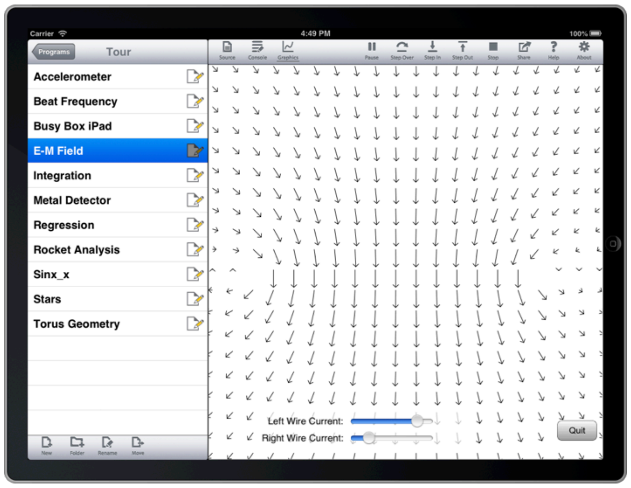
Beat Frequency

This shows an overlay of two sine waves with different frequencies. It's a good example of multiple functions plotted at once, as well as showing a lot of various options for axis titles, color choices and line styles. This sample also shows how to mix controls with plots. The two sliders at the bottom of the screen control the frequency, letting you explore different frequencies and how they overlap. You can stop the program with the Stop button on the button bar, but you can also stop it with the Quit button that appears to the right of the sliders.

Busy Box

techBASIC gives you easy access to the controls available on an iPhone or iPad. This sample shows exactly how to use most of the controls. The sample is also the star of one of the many tutorial blogs on the Byte Works web site, where you will find a detailed walk through of the program describing all of the steps needed to create each of the controls.





E-M Field

Two infinite length wires with a current running through them will create a magnetic field. This program shows that magnetic field, and lets you vary the current through each wire from -2 amps to 2 amps. As you vary the current, the vector field updates. Press the Quit button to exit the program.

This program shows how to visualize vector fields, like electromagnetic fields and gravity. Vectors display moth direction and magnitude. They are also available in 3D plots.

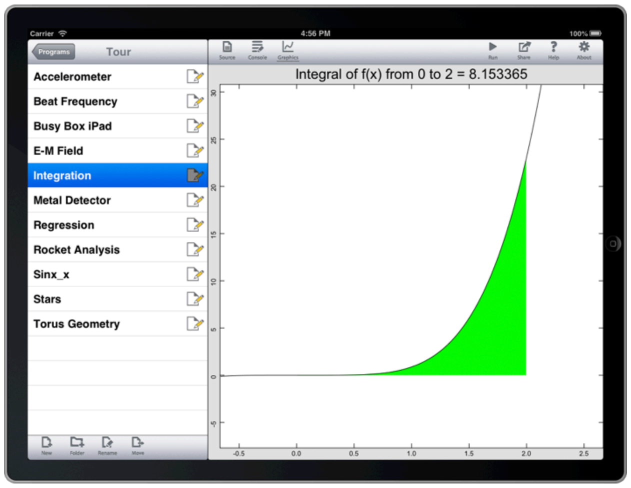
Integration

Finds the value of

$$\int_0^2 x^4 \log\left(x + \sqrt{x^2+1}\right) dx$$

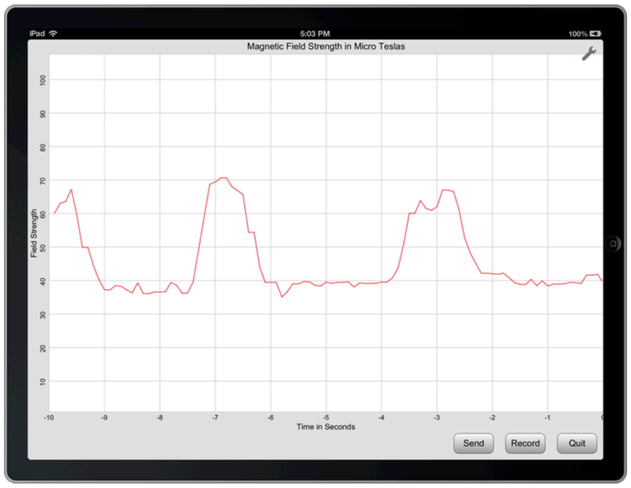
using the trapezoid rule. It's an example of one of the powerful built-in math functions in techBASIC. Calculation of the integral is done with the single line

```
    ans = Math.trap(FUNCTION f, x0,
x1)
```



It’s easy to change the limits of integration or the function that is integrated, making this a great calculator for Calculus homework.

See the tutorial blog *Numeric Integration with techBASIC* on the Byte Works web site (www.byteworks.us) for a complete discussion of this sample.



Metal Detector

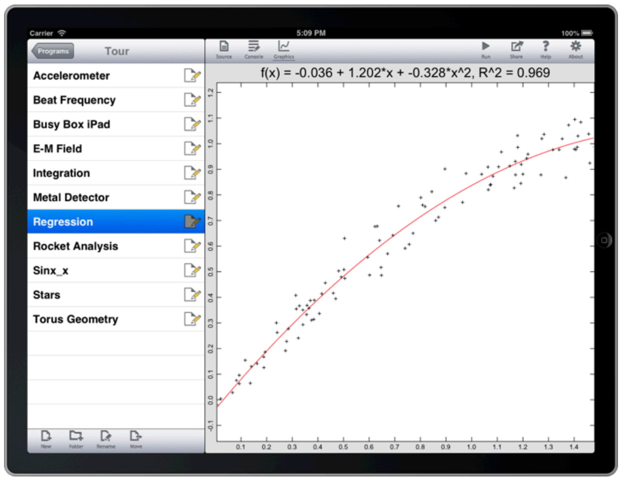
Here's a metal detector created from the magnetometer in an iPad, shown as it passes back and forth over the edge of an aluminum keyboard. It's a great example of practical applications that can be built from internal sensors. It's also the subject of a chapter in the O'Reilly book, *Building iPhone & iPad Electronics Projects*.

Regression

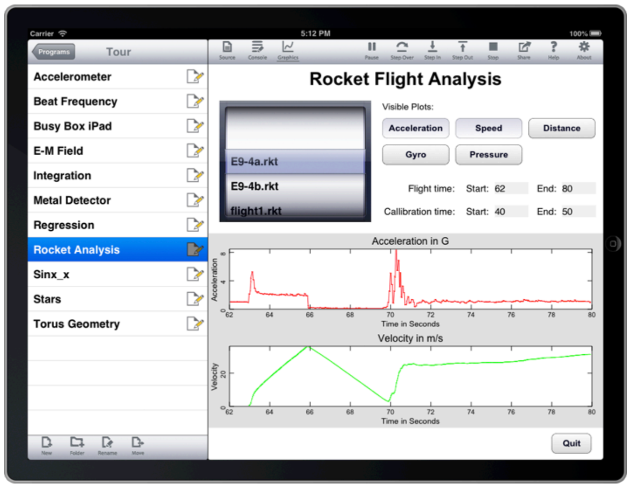
Performs a second order polynomial fit using linear regression. The results are plotted, along with the data points, and the fit is displayed as the plot title. The goodness of fit, R^2, is also calculated and shown in the plot title. As with Integration, the heavy lifting to perform the regression analysis is done with a single call to a built-in math routine.

It’s easy to change this sample to fit polynomials of different orders, or to fit any data file with comma separated values.

See the tutorial *Linear Regression with techBASIC* on the Byte Works web site (www.byteworks.us) for a complete discussion of this sample.



f(x) = -0.036 + 1.202*x + -0.328*x^2, R^2 = 0.969



Flight time: Start: 62 End: 80
Calibration time: Start: 40 End: 50

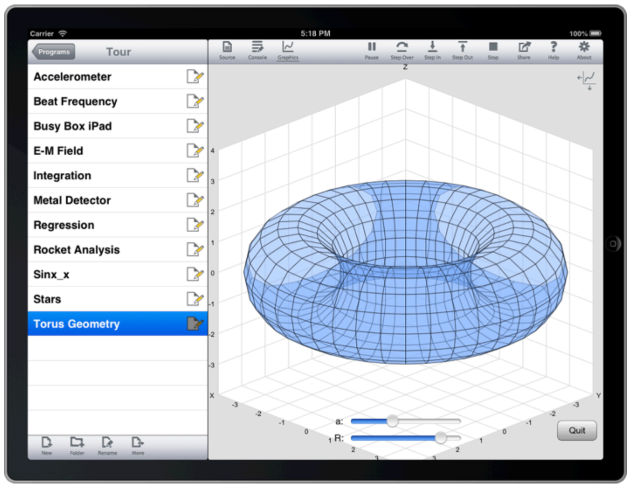
Rocket Analysis

Did you ever wonder what would happen if you tucked an iPhone into the payload bay of a model rocket? Rocket Analysis shows you, displaying the results of six actual rocket flights tracked by a Texas Instruments SensorTag. This Bluetooth LE device broadcast data back to an iPhone on the ground on some flights, and to an iPhone carried with it in the rocket on others. Get the full story on the Byte Works web site or in the O'Reilly book, *Building iPhone & iPad Electronics Projects*. Be sure and check out the movie showing the rocket flights!

Torus Geometry

Sometimes data doesn't fit the neat mathematical definition of a function. techBASIC can still plot the information, as this program shows. It creates a sheet of point-cloud data, then plots the data in three dimensions.

The controls put this to use, allowing you to explore the relationship between the diameter and thickness of the torus. It's a great example of visualizing a complex topic from descriptive geometry. It's also a lot of fun!



The folders of programs are divided up by topic. Many programs make sense in more than one topic, so they appear multiple times as aliases. Changing the program in one place changes it everywhere.

Here is your guide to the folders. It will help you find the programs that interest you the most.

Tour

You've seen this one, It has a collection of programs to show you some of the major features in techBASIC.

Bluetooth LE

techBASIC can communicate with any Bluetooth LE device. This folder has a number of programs that show how it's done. All of the programs require specialized hardware, since you need the Bluetooth LE device to use the program.

Programs in this folder appear in either the blogs section of the Byte Works web site at www.byteworks.us , or the O'Reilly book, *Building iPhone & iPad Electronics Projects*, or both.

Byte Works Blogs

The Byte Works web site, www.byteworks.us, has a collection of tutorials on topics ranging from flying an iPhone on a model rocket to numerical integration. The source code for all of the blogs is in this folder.

Graphics

You've seen some of the powerful graphics capabilities in techBASIC. This folder runs through the major capabilities, showing things like cylindrical coordinates, polar coordinates, and plotting sheets of X-Y-Z data.

HiJack

HiJack is a device that plugs right into the headphone port on an iPhone. It's an 8 bit A-D converter than can convert a resistance into a digital value. techBASIC makes it easy to read values from HiJack so you can build all sorts of devices, like a plant moisture meter.

The programs in this section appear in both the Byte Works web site at www.byteworks.us and the O'Reilly book, *Building iPhone & iPad Electronics Projects*. They require a HiJack device to work.

Math

techBASIC is a powerful technical computing environment. This folder has several examples that show how easy it is to do rather sophisticated calculations in techBASIC.

My Programs

This folder is empty. It's reserved for all of your own great programs!

O'Reilly Books

All of the samples from the O'Reilly book, *Building iPhone & iPad Electronics Projects*, appear in this folder. You don't have to type them in, and you can trace through them in the debugger to see how they work!

You'll find programs to use all of the sensors in the iPad, connect and use HiJack, and access Bluetooth LE devices. The book shows how to build moisture meters, metal detectors, model rocket accelerometers and more.

Programming

Here are some utilities for programming, like Dump, which shows the contents of any file. You'll also find programming examples, like the Busy Box sample discussed earlier.

Science

From simulations, like the classic Slime Mold self-organizing system, to data collection, such as the Gyroscope, this is a folder for all of the science and engineering folks.

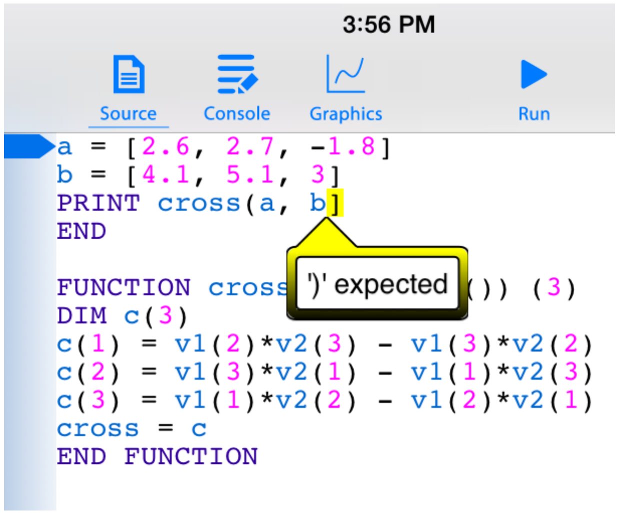
Debugger

Every programmer knows Lubarsky's Rule of Cybernetic Entomology, even if they are not familiar with the name: There's always one more bug. techBASIC has several features to help you tame cybernetic arthropods.

Error Messages

If there is a compile error in your program, or if techBASIC detects a problem while the program is running, an error message is shown in the Source view. The approximate location is highlighted, and an error message describes the problem. Tap the view to dismiss the error message. You can tap the highlight to see the message again, but once you start editing, all of the messages are reset.

See Appendix A of the reference manual for more details about specific errors.

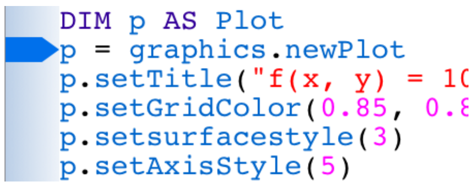


```
a = [2.6, 2.7, -1.8]
b = [4.1, 5.1, 3]
PRINT cross(a, b]
END

FUNCTION cross                    ()) (3)
DIM c(3)
c(1) = v1(2)*v2(3) - v1(3)*v2(2)
c(2) = v1(3)*v2(1) - v1(1)*v2(3)
c(3) = v1(1)*v2(2) - v1(2)*v2(1)
cross = c
END FUNCTION
```

')' expected

Setting and Clearing Breakpoints

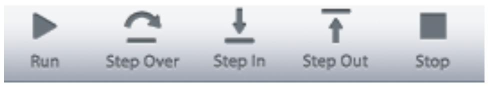


```
DIM p AS Plot
p = graphics.newPlot
p.setTitle("f(x, y) = 1(
p.setGridColor(0.85, 0.8
p.setsurfacestyle(3)
p.setAxisStyle(5)
```

Setting a breakpoint lets you stop a program at a specific location, examining the current values for variables and watching them change as you step line-by-line through the program. The blue bar to the left of the Source view is the breakpoint bar. Tap on the breakpoint bar to set a breakpoint; tap again to clear it. A line with a breakpoint displays a blue marker like the one shown here.

With a breakpoint set, running the program will stop it at the first breakpoint encountered.

Debugging Programs

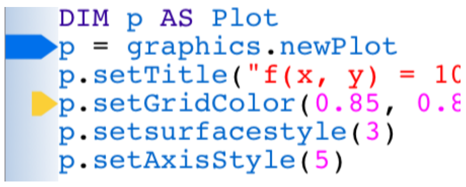


There is a run button on the button bar that can run the currently selected program. While the program is running, this button is replaced by a series of buttons. These are used to pause, cancel or debug the running program. Let's step through these and see what they do.

The Pause Button

Many programs execute so fast that you never see the debugger buttons, but if a program is taking a while, you may notice that the Run button is replaced by a Pause button. Tap the Pause button to stop the program at the currently executing statement.

Stepping Through Programs



```
DIM p AS Plot
p = graphics.newPlot
p.setTitle("f(x, y) = 1(
p.setGridColor(0.85, 0.8
p.setsurfacestyle(3)
p.setAxisStyle(5)
```

Once a program is stopped at a breakpoint or with the Pause button, you can step through the program one line at a time to see the program flow. The currently executing line is marked by a gold triangle, like the one seen here after stepping two lines past the breakpoint.

Step Over steps one line at a time, even if the line being executed calls a subroutine or function that executes many lines before returning.

Step In works just like Step Over on lines that don't call a subroutine or function, but if the line does make a call, execution pauses at the first line of the subroutine or function.

Step Out is generally used from inside a subroutine or function. The program runs at full speed until the subroutine or function ends, then pauses at the first line encountered after returning. If Step Out is used from the main program, the program runs at full speed to completion.

In all cases, the program will stop if a breakpoint is encountered. For example, stepping over a subroutine with a breakpoint will still stop at the breakpoint.

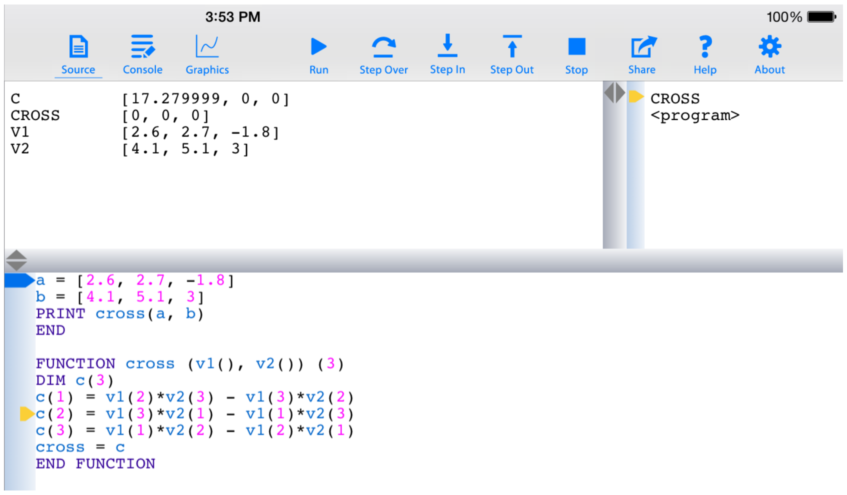
Resuming Full Speed Execution

Once a program is stopped, either by tapping Pause or by encountering a breakpoint, tapping the Run button causes the program to continue execution from the current location. It will run at full speed until the Pause button is pressed, a breakpoint is encountered, or the program completes.

Stopping a Program

Use the Stop button to immediately stop execution of a running program. This works whether the program is stopped by the Pause button or a breakpoint, or running at full speed. An error message will be displayed at the line that was executing when the Stop button was pressed.

The Variable View



```
C            [17.279999, 0, 0]
CROSS        [0, 0, 0]
V1           [2.6, 2.7, -1.8]
V2           [4.1, 5.1, 3]
```

```
CROSS
<program>
```

```
a = [2.6, 2.7, -1.8]
b = [4.1, 5.1, 3]
PRINT cross(a, b)
END

FUNCTION cross (v1(), v2()) (3)
DIM c(3)
c(1) = v1(2)*v2(3) - v1(3)*v2(2)
c(2) = v1(3)*v2(1) - v1(1)*v2(3)
c(3) = v1(1)*v2(2) - v1(2)*v2(1)
cross = c
END FUNCTION
```

When a program is stopped, two new subviews show up in the Source view. The view at the top left is the Variable view. It shows all of the variables in the current program scope; that is, all of the ones in the current subroutine, function, or the main program.

In the sample shown here, the program is currently in a function called cross, computing the cross product of two vectors. There are four variables in the function. The first is `C`, a matrix used to hold the result inside the function. The first element of `C` has been set to 17.279999, while the last two have yet to be changed from the initial values of 0. `CROSS` is the name of the function itself, showing the value that will be returned to the caller. The last two variables are the parameters that were passed to the function. These values change as the program steps.

There are times when the variable view is not wide enough to show the complete value of a variable. There are two ways to deal with this. The first is to slide the gray divider to the right, increasing the width of the variable view. The second is to tap the variable, which brings up a detail window showing the full value of the variable. This still may not provide enough space for a really large array or very long string, but you can scroll the value in the detail view to see the rest of the value.

The Variable view can be scrolled to show additional variables if there are too many to show at one time. Dragging the horizontal gray divider will also increase or decrease the height of the Variable view as needed.

The Stack View

The Stack view appears immediately to the right of the Variable view. It shows the current call stack. The program itself always appears at the bottom of the list. Names of new subroutines and functions are pushed onto the top of the stack as they are called, and popped off of the list as they return.

The gold triangle points to the current scope. The current scope is the one whose variables are shown in the Variable view, not necessarily the one that is executing. Tap the name of a scope to change the current scope. For example, tapping `<program>` in the sample views shown will change the scope to the program, displaying the values of A and B.

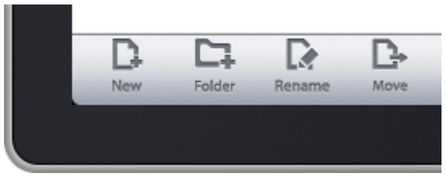
The Full Screen Tools Menu

It is possible to claim the entire screen for a techBASIC program. A normally functioning full screen program should have some way to stop itself, generally by using a Quit button, but what if the program has a bug? How do you stop it or debug the program?

Whenever a program is running in full screen mode, techBASIC adds a special button shaped like a wrench to the top right of the screen. Tap this button and a list of debug button appears, allowing you to access all of the debug functions discussed above even though the standard button bar is hidden.

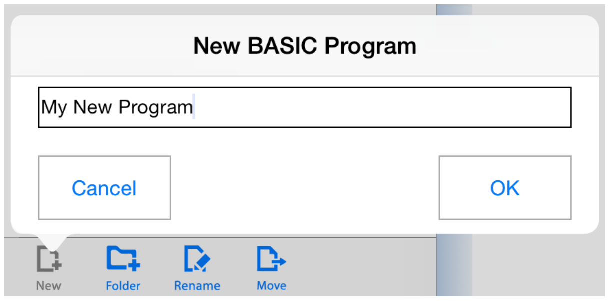
There are also a couple of new buttons. Send to Photo will copy the current screen to your photo library, where is can be manipulated or shared with others. Pan/Zoom Graph is used to switch between panning and zooming a graph or the axis of the graph.

Manipulating Programs



The row of buttons at the bottom of the program list is used to create and manage programs and folders.

Creating Your Own Programs



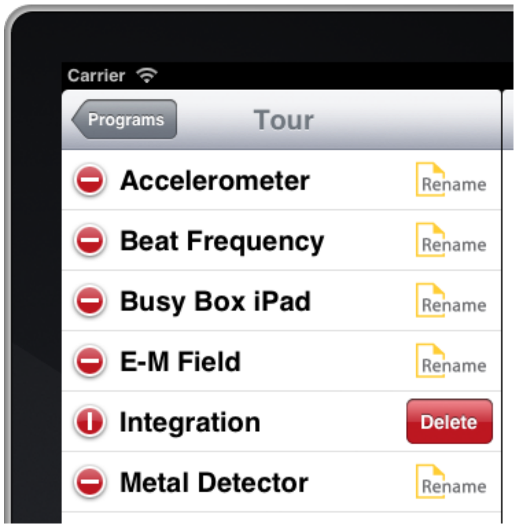
Tap the New button at the bottom of the program list to create a new program. A dialog like the one shown appears. Enter the name of the new program and tap OK. The new program is created and opened in the console view.

The new program is created in whatever folder is open at the time.

Creating Folders

The Folder button to the right of New is used to create a new folder. Just as with New, a dialog opens, allowing you to enter the name of the new folder. Also like New, the new folder is created in whatever folder is open at the time.

Renaming and Deleting Programs



Tap the Rename button at the bottom of the Programs view to delete or rename an existing program. This changes the program view as shown to the right, adding delete dials and rename buttons.

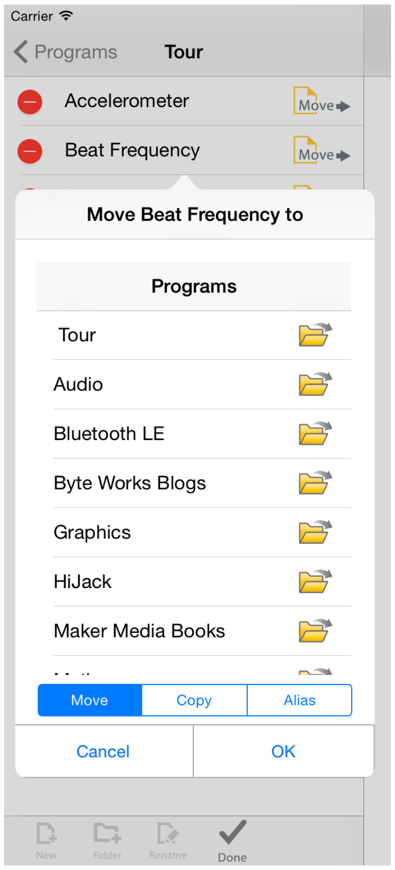
Tapping a delete dial rotates it sideways and presents a Delete button to the right. Tapping that button deletes the program. There is no way to undo deleting a program, so be sure you really want to delete it before confirming the delete.

Deleting sample programs is a special case. Deleting a sample program definitely deletes the sample, but it can be recovered. See *Preferences*, later in this chapter, for an option to recover deleted or edited samples.

There is also a shortcut for deleting a program. Before you even tap the Edit button, swipe left across the program you want to delete. This presents the Delete button right away, skipping the steps of tapping Edit and the delete dial.

The other option while editing is to rename a program or folder. Tap the Rename button, and you will see a dialog showing the current name of the program. Change the name to whatever name you prefer and tap OK to rename it. Programs are always alphabetized, so look for the program in its new location.

Moving Programs



Tapping the Move button changes the program list to expose a Move button, as well as the same Delete button available when renaming programs and folders. Tapping one of the Move buttons to the right of a program name brings up a dialog allowing the program to be moved, copied, or aliased.

Start by selecting the new location for the program by navigating to the proper folder in the Programs list. This works just like navigating folders in the standard Programs list, but only folders are listed.

Select Move, Copy or Append. Move will move the file from its current location to the new one. Copy creates a new working copy of the program, while Alias creates an alias of the original. They look similar, but there is a key difference. A copy of a program is a new instance of the program. Changing the program does not change the original. An alias is just a name for the original program in a new place. Changing the alias will change the original program, too. Aliases are used extensively in the sample programs, allowing the same program to appear in different topical folders. For example, the Accelerometer appears in the Tour, O'Reilly Books and Science folders. Two of these are aliases, so the program itself only has to exist in one place.

Tap OK to move, copy or alias the program, or Cancel to exit without making a change.

Sharing Programs and Graphs

All three of the views have a share button. This is one way to get information from techBASIC to other programs or people.

The Share button on the Source and Console views share any text currently in that view. It creates a new email message containing the source code or console text, ready for the user to address and send.

What if you get a program by e-mail, and want to use it in techBASIC? Use the iPad's copy command to select and copy all of the text in the e-mail, create a new program in techBASIC, and paste it.

The Share button on the Graphics view is a bit different. The current Graphics view is redrawn at a new resolution and sent to the Photo app. From there, you can e-mail it, leave it there in a collection to show others, or perform any of the manipulations supported by the photo app. By default, the plot that appears in the Photo app is about a megapixel. You can change this in the preferences, generating up to a 4-megapixel image. See the reference manual if you are not familiar with using preferences; it will walk you through all of the techBASIC preferences.

Use a Keyboard!

Some folks find the digital keyboard built into the iPad is perfectly adequate for working with text. For short tasks, I agree, but for really long ones, I do like a physical keyboard.

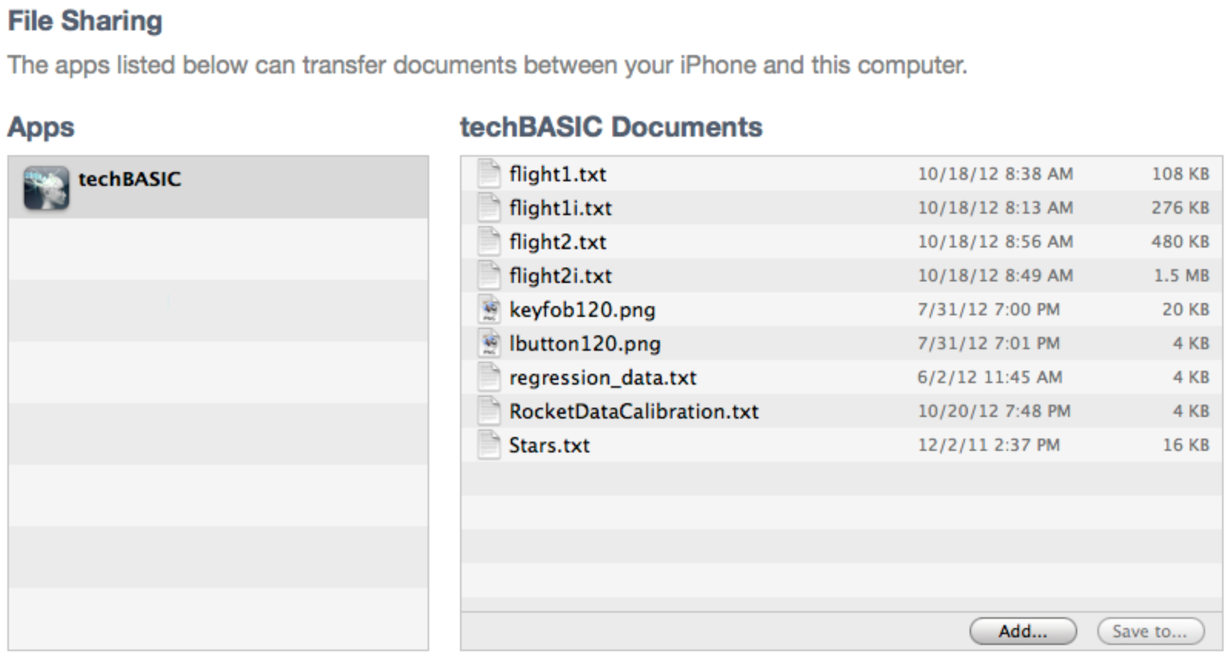
Fortunately, the iPad supports a Bluetooth keyboard, and it works great with techBASIC. In addition to the obvious utility, be sure and try holding the shift key down while you use arrow keys to move the cursor. This selects text. The standard keyboard shortcuts for manipulating sections also work:

⌘C	Copy
⌘V	Paste
⌘X	Cut
⌘F	Open the Find dialog
⌘G	Find the next occurrence of the find string (the Find dialog is not opened)
⌘]	Indent (adds two spaces to the start of each selected line)
⌘[	Outdent (removes up to two spaces from the start of each selected line)
⌘→	Move to the end of the line
⌘←	Move to the start of the line
⌘↑	Move to the start of the file
⌘↓	Move to the end of the file
⌘-shift→	Select to the end of the line
⌘-shift ←	Select to the start of the line
⌘-shift ↑	Select to the start of the file
⌘-shift ↓	Select to the end of the file
option→	Move to the end of the word or number
option ←	Move to the start of the word or number
option ↑	Move to the start of the line, or the previous line if at the end of a line
option ↓	Move to the end of the line, or the next line if at the end of a line
option-shift→	Select to the end of the word or number
option-shift ←	Select to the start of the word or number
option-shift ↑	Select to the start of the line, or the previous line if at the end of a line
option-shift ↓	Select to the end of the line, or the next line if at the end of a line

Moving Programs and Data To Your Desktop Computer

There are many reasons to want to move a program or data from your iOS device to your desktop computer. You may want to share data with a colleague, use a program in another implementation of BASIC, or perhaps just edit a long program in a traditional keyboard and mouse environment.

Data files are stored in an area called the sandbox. iTunes can be used to move these files to and from your desktop computer. To move a data file, start iTunes and select your iPad from the list of devices. You will see a list of tabs across the top of the device window; select the tab named Apps. Near the bottom of the window, you will see a list of applications that can share data. Select techBASIC, and you will see a list of all data files.


File Sharing

The apps listed below can transfer documents between your iPhone and this computer.

Apps

techBASIC

techBASIC Documents

flight1.txt	10/18/12 8:38 AM	108 KB
flight1i.txt	10/18/12 8:13 AM	276 KB
flight2.txt	10/18/12 8:56 AM	480 KB
flight2i.txt	10/18/12 8:49 AM	1.5 MB
keyfob120.png	7/31/12 7:00 PM	20 KB
lbutton120.png	7/31/12 7:01 PM	4 KB
regression_data.txt	6/2/12 11:45 AM	4 KB
RocketDataCalibration.txt	10/20/12 7:48 PM	4 KB
Stars.txt	12/2/11 2:37 PM	16 KB

Data files are created by your programs or dragged into the sandbox using iTunes. Presumably you know what is in the file and what other programs can edit the file. If you are unsure of the contents of a file, use the Dump sample in techBASIC to view the contents, or check the techBASIC reference manual for details on file formats.

With the documents window open, the easiest way to move files to and from the iPad sandbox is to drag the file in or out of the document area. Traditional Open and Save dialogs are also available from the Add... and Save to… buttons. To remove a file, select the file and press the delete key on your keyboard.

techBASIC source files are not stored in the sandbox. This is due to a restriction placed on programs not written by Apple or by Apple subsidiaries. Except for Apple programs, no program is allowed to move executable code to and from an iOS device using iTunes. As a result, moving programs to and from techBASIC is considerably more cumbersome than moving data files.

To move a program from another device to your iPad:

• Place the source code in an otherwise blank email and send it to yourself.
• Read the email on the iPad.
• Tap twice on the email, then tap the Select All button to select all of the text.
• Tap the Copy button to copy the program.
• From techBASIC, tap the New button on the Programs list and enter a new program name.
• Paste the source into the new program.

To move a program from your iPad to another location:

• Display the source code by tapping the Source button.
• Tap the Share button. This places the contents of the source file in an email that you can send to yourself.
• Read the email on any other device and copy and paste the source to save it to the new device.

We apologize for the difficulty in performing this seemingly simple task. We will immediately add the ability to move programs to and from an iPad using iTunes (or through any other means) should Apple ever allow us to do so.

Resources

This Quick Start Guide is designed to show you generally what techBASIC can do, and introduce you to features and ideas that may not be obvious to people who have used other programming environments. It's a start, but hardly all you need to use the program well. Here are some resources that, along with your own curiosity and creativity, will help you get the most from techBASIC.

Reference Manual

A reference manual is to a programmer what a dictionary is to a writer. Every good writer has and uses some kind of dictionary, but none of them learned to write from one.

You should download and review the techBASIC Reference Manual, available from the Byte Works web site at www.byteworks.us. Depending on your background and interest, you will spend more time in some sections than others. Pretty much everyone should skim Chapter 2, which describes the user interface for the iPad. Browse a bit in Chapters 4 through 12, which describes the techBASIC programming language, so you know where to find information when you need it. If you are unfamiliar with arrays in BASIC, and how they can be used to manipulate matrices, read the sections on arrays in Chapters 4, 7, 8 and 12. Carefully review Chapters 13 to 17, which presents the unique features of techBASIC that let you collect and plot information, create controls, and handle events in your programs.

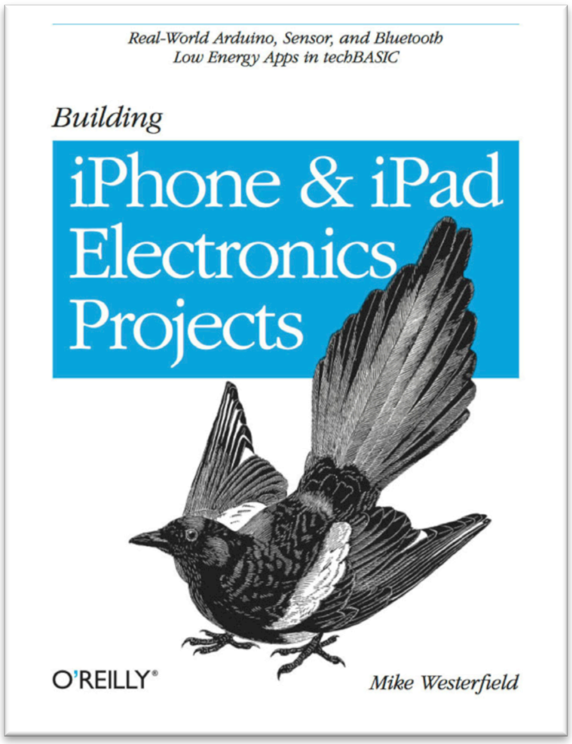
Tutorials and Blogs

The Byte Works web site has a growing collection of tutorials and articles in the Blogs section. Visit the blog page at www.byteworks.us. Let us know of any topics you would like to see covered by dropping us a note at support@byteworks.us. Be sure and get in touch if you are interested in writing a guest blog, too!

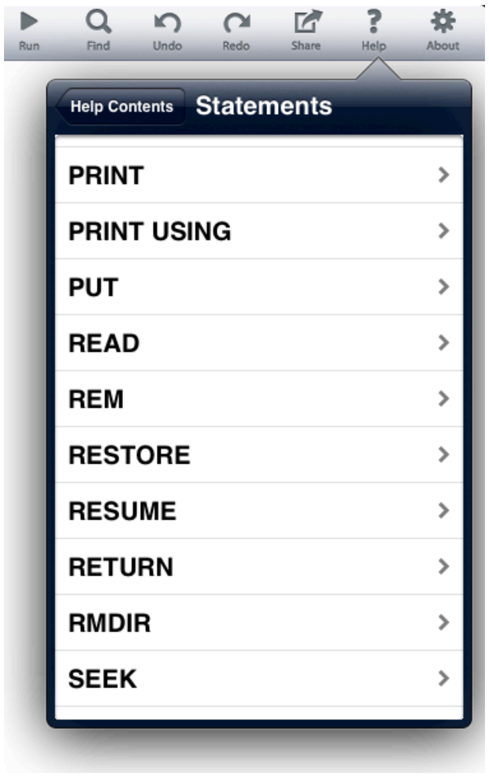
Building iPhone & iPad Electronics Projects

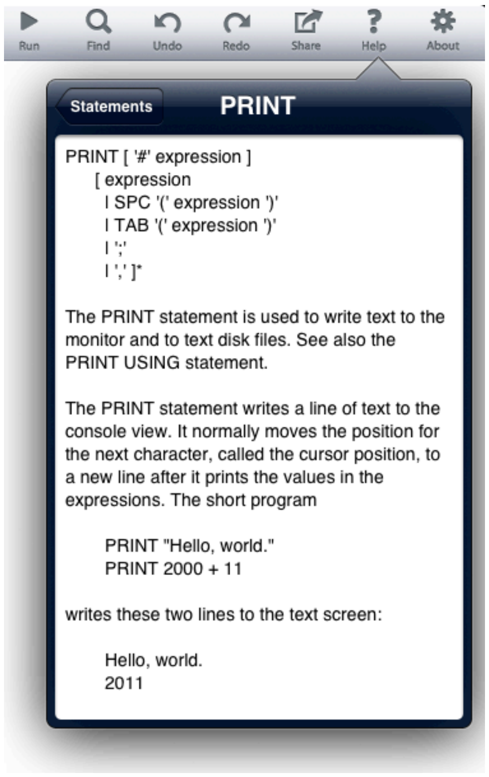
While techBASIC is a general purpose programming language, capable of writing any type of program, it's optimized for technical programming. Key parts of technical programming are accessing sensors, communicating with TCP/IP, communications with Bluetooth LE, and using devices like HiJack.

This new book from O'Reilly Media covers all of these topics with fun projects like building a moisture meter with HiJack or collecting data from a model rocket flight. It's available now as a prerelease electronic book, and is due out in print form in September 2013.


Real-World Arduino, Sensor, and Bluetooth Low Energy Apps in techBASIC
Building
iPhone & iPad Electronics Projects
O'REILLY
Mike Westerfield

Help





```
PRINT [ '#' expression ]
     [ expression
       | SPC '(' expression ')'
       | TAB '(' expression ')'
       | ';'
       | ',' ]*
```

The PRINT statement is used to write text to the monitor and to text disk files. See also the PRINT USING statement.

The PRINT statement writes a line of text to the console view. It normally moves the position for the next character, called the cursor position, to a new line after it prints the values in the expressions. The short program

```
PRINT "Hello, world."
PRINT 2000 + 11
```

writes these two lines to the text screen:

```
Hello, world.
2011
```

Even programmers who use a language forty (or more!) hours a week don't remember every command, or every detail about every command, in the languages they use. The Help facility gives you all the technical details about every statement, function and class in techBASIC. Tap on a topic to expand on the topic, or use the back button at the top of the dialog to move back a level.

While Help has listings for all of the language, it does not include figures, samples or information about the user interface. Help is a great quick reference while writing a program, but it's not a substitute for the techBASIC Reference Manual.

Support

Lubarsky's Rule of Cybernetic Entomology applies to techBASIC, too. If you find a bug, please let us know. You can send bug reports to support@byteworks.us. We will make every effort to squash them as they are reported.

You may need a feature that is not in techBASIC. Send suggestions to the same support address. While we will not be able to implement every suggestion from everyone who uses techBASIC—experience even shows that some will be contradictory—we will listen to all of your suggestions and consider them carefully. The more people who ask for a specific feature, the more likely it is to be implemented, so drop us your wish list even if you know a colleague has already sent us a similar request.

You may just need some assistance with the program. While we cannot offer free custom programming or debugging help, we can and will attempt to help you if you have questions about the program.

If you would like to become an active part of the community of techBASIC programmers, join us on Facebook at http://www.facebook.com/ByteWorks, where you can post your own programs, ask questions, and find out about new product information and outside resources like new blogs on the Byte Works' blog page at http://www.byteworks.us/Byte_Works/Blog/Blog.html.

Finally, check our web site periodically for blogs, new support channels, new companion products and update information.