## Release Notes (3.17, September 2018)

The calculator now lets you pick from a couple of fonts including two antique digital fonts.

The Editor has changed dramatically! Users may know that the editor used to be OK for short programs, but had trouble keeping up when a large program is present. An entirely new editor has been created for BC BASIC. You will note that the speed is greatly enhanced and there are powerful new capabilities:

* There’s a FIND and GO TO LINE ability
* You can INDENT code easily

In addition, the underlying “compiler” has been completely rewritten. It’s now very substantially faster (the 2,000-line Wumpus game used to tax it severely; the same game compiles almost instantly in the new compiler). Although much effort was spent to ensure that programs compile and work the same way, it’s likely that there will be a few changes that snuck through. Please accept my apologies in advance if this cause you any problems.

There are a TON more example programs including some fun game and math programs.

Setting colors is more consistent; color=<int> works for everything.

The INPUT expression used to require zero or one for DEFAULT and PROMPT, and they had to be in the order DEFAULT and then PROMPT. This is no longer required.

The PLAY command will PLAY STOP more quickly. In the past, PLAY STOP would wait until the current note stopped; now the note is stopped almost right away. This was important for the Hunt the Wumpus game, where the “pit” and “monster” sounds are drawn-out single notes (e.g., long ‘ocean’ notes). This prevented the user from making a move right away; the sound effects had to complete before the note could be played.

Arrays can be two-dimensional now! You can either make a 2-D array from the start with DIM or you can build one up.

Example making a 2-D array with DIM

REM Make an array with 2 rows and 5 cols   
DIM array(2,5)   
array.Fill (0)  
array[r,c] = 1  
PRINT array[r,c]

Example making a 2-D array with AddRow  
  
REM Make an array with 2 rows and 5 cols  
DIM array()  
array.AddRow (11,12,13,14,15)  
array.AddRow(21, 22, 23, 24, 25)  
PRINT array[r,c]

The array.Add(<value>) method can now take multiple values. A call like array.Add (1,2,3) is just like calling

Array.Add (1, 2, 3)

REM The old way is more work!

array.Add (1)  
array.Add (2)  
array.Add (3)

There’s a new array.Fill ( value) method that will entirely fill every element of a 1- or 2-dimensional array with the same value. The most common use is to zero-fill an entire array.

Common example:  
DIM array(2,5)  
array.Fill (0.0)

INPUT can now read in multiple values at once. INPUT X, Y, Z will read three items from the user into the X, Y and Z variables.

READ can now read in multiple values at once. READ X, Y, Z will read three DATA items into the X, Y and Z variables.

Added RESTORE to reset the READ/DATA statements. READ normally starts at the first item in the first DATA statement and each subsequent READ picks the next item. RESTORE sets READ so that it will read from the first item again.

Added RND() as a function in addition to the normal RND value. This makes BC BASIC more compatible with other BASIC versions. RND() with no parameters will return a random number between 0 and 1, as will RND(positive number). The arguments isn’t used. RND(0) will reset the random number generator in a random way, and RND(negative number) will reset the random number generator using the given seed.

Added SPC(n) function to improve compatibility with other version of BASIC. The SPC(n) function will return a string with <n> spaces (rounded down if needed). It’s often used in conjunction with PRINT as in PRINT SPC(5); “Hello” to indent the Hello.

Added graphic LineTo(x,y) and GoTo(x,y). These make drawing some types of graphs much easier because your code doesn’t have to keep track of the old x,y position.

The graphics g.Circle (x, y, radius) now has a get and set Radius property to change the radius of a circle.

Graphics now includes a set of Scaling methods to create a window on the screen, set the scale of the window, and to use different scaling modes. This is commonly used when creating physical-based displays: you can draw and move your graphs based on physical units like “centimeters” and have them automatically scale correctly. Look at the Bubble Animation demo for a good use of this ability.

Added bitwise operators for the Math object: Math.BitAnd (left, right), Math.BitOr (left, right) and Math.BitNot (value). These do bitwise operations. For example, PRINT Math.BitAnd (0x6, 0x3) will print 2. 0x6 is bit 0110 and 0x3 is 0011. Their AND is 0010.

The String.Parse (“csv”, “1,2,3”) method used to return only string type values. It will now attempt to convert each input to a double; if the conversion works it’s kept. The old way was a surprise especially when comparing the results; if data[1] has an actual value of 7, IF (data[1] > 200) THEN PRINT “BIG” would actually print BIG! That’s because BC BASIC would decide that the compare should be done as strings, and the string “7” is bigger than the string “200”. This cost me five wasted hours of debugging during a recent Hackathon!

Added String.Pos (string, lookFor, startingIndex) to search for a value inside a string starting at a given starting index.

Added String.Replace (string, startingIndex, length, replacement). The portions of the input string from startingIndex for the given length will be replaced with the given replacement. The size of the string to be replaced is entirely unrelated to the string replacement; you can “squeeze” a string to be smaller, or you can “expand” a string. The length to be replaced can have a length of zero, in which case the replacement string is spliced in.

This method is provided to make it easier to port programs from systems like the Tektronix 4050 series.

System. FolderBasic returns the folder that BASIC apps are saved to. System.FolderTemporary is a folder where time-stamped BASIC programs are saved.

System.Trace(0) works now; it had accidently been disabled. System.Trace(1) will cause some level of tracing in your program (more usefully, it says how many callbacks were suppressed and the number of lines/second are being evaluated). System.Trace(0) (the default) should suppress all tracing.

The TI SensorTag 1350 (and other TI Sensor Tags) are now manufactured without an IR temperature chip. TI’s [website](http://processors.wiki.ti.com/index.php/CC2650_SensorTag_User's_Guide#IR_Temperature_Sensor) says that they are no longer in the contactless IR sensor business, and no longer have any TMP007 in stock. The new boards have a blank solder pad where the chips would be. This condition is not readily detectable; you can still try to enable the IR Sensor, but you will not get any IR callbacks.

The Unicode list has been updated to Unicode version 11. New characters include OVERLAPPING WHITE AND BLACK SQUARES , the COPYLEFT SYMBOL, LAB COAT and SUPERVILLAIN.

**Important bugs fixed:**

It’s perfectly OK to make a nested (recursive) array. But in older version of BC BASIC, this would cause an instant crash in BC BASIC! The array code has been fixed; now when an array is printed and one of the values includes a sub-array, the sub-array printing is short-circuited,. The length of the sub-array is printed in superscripted number.

The test for this fix looks like the following:

FUNCTION TEST\_NestedRecursiveArray()

DIM a(3)

a(1) = 1.1

a(3) = 3.3

a[2] = a

string = ""+a

ASSERT (string = "[1.1,[³],3.3]")

RETURN

When the array ‘a’ is converted to a string, it’s converted as [1.1,[³],3.3]. The array in the middle is replaced with [³], indicating that it’s an array that’s three elements long.

Other bugs fixed:

The CLS command didn’t accept two numeric color values (like CLS 1 2). This now works.