# Adding new statements to BC BASIC

There are two steps to adding new statements to BC BASIC. The first is to update the *parser* so that BC BASIC accepts the new statement. The second is to hook up a class to the parser so that the statement, when run, actually performs some action.

The BASIC source code is in the BCBASIC directory. Inside that directory is a “BCBasic.tpg” file that defines what the language (the LET a=b commands) look like. The TPG file is opened with the program TinyPG, the Tiny Parser Generator.

To add the PLAY <string> statement, I did the following:

1. In the TPG file, add the definition of the PLAY <string> command
   1. Add a PLAY reserved word PLAY -> @"PLAY"; at about line 27. This creates PLAY as a reserved word.
   2. Add a PLAY <string> statement Play\_Statement -> PLAY Expression { return new BCBasic.Play ($Expression as BCBasic.IExpression); }; at about line 212. This creates PLAY as a statement that takes a single parameter that must be an expression, and tell TinyPG that when a PLAY statement is seen to create an object of the “Play” class. We haven’t made the Play class yet; that will happen in step two
   3. Add the Play\_Statement to the list of statements at about line 246. This means that the Play\_Statement will be accepted just like all the other common statements.  
      Statement -> Call\_Statement | … | Play\_Statement | Print\_Statement ..
   4. ^S to save the file
   5. Build🡪Generate to make the new C# code that Best Calculator uses as a parser
   6. Exit TinyPG; you are done with the .TPG file
2. In the Best Calculator solution, create the Play command
   1. All of the statement classes (like Print) are defined in BCBasic.CS. Open that file; there are a whole series of classes. PLAY is pretty similar to RAND (they each just take a single expression), so that’s my starting point
   2. I picked AdvancedCalculator as a place to add PLAY. AdvancedCalculator is where “most” of the code lives, and BluetoothCalculatorUniversal then shares it. Things added to AdvancedCalculator will show up in Bluetooth (IOT) calculator.
   3. I started by copying the Rand class and renaming to Play. Keep the statement classes in alphabetical order; it will make your life easier!
   4. I already had a class that implemented the Music Macro Language, which is the music that PLAY can do. I added it to the RunTimeLibrary folder
   5. Now in the Bluetooth project, also add the files, but add them from the AdvancedCalcualtor area, and add as links. This sets up the files so that changing the files in one place will also change them in the other.
3. Create samples’
   1. Create a new Package call EX: PLAY music and sounds and add in some nice samples. Be sure that there is documentation for the package and each program.
   2. Add the package to the Assest libraries. Just like before, add the actual file to the AdvancedCalculatorUniversal and a link in the Bluetooth version. Make the file by exporting the package. Add the files as Content and Copy If Newer
4. Document
   1. Update the Addition document!
   2. Is this the last change before shipping? If so, update the real document and make a PDF. There’s a bunch of steps in the “how to ship” guide.