Moving to Modern Programming: BBjDE

by Nick Decker

BBx® has always been known as a language that was easy to learn and easy to program. With the introduction of BBj™, BASIS will make the language easy to program *in*. Editing, running and debugging BBx applications will be dramatically improved by BBj's use of contemporary graphical controls in its integrated development environment (IDE).

BBjDE™ will allow faster, more efficient coding as it removes some of the cumbersome restrictions that hindered programmers in the past. It will also offer new features that encourage the writing of code that is better organized and more legible. Put simply, the BBjDE will change the way you write and debug your applications. Of course, it will still support the older coding methods to which many BBx developers are accustomed, but once you begin to take advantage of its debugging features you'll be hooked.

For example, it used to be somewhat difficult to view your program. The LIST command was used to show screenfuls of code, but it was difficult to show just the portion of code that you wanted. To keep the code from running off the screen, line numbers or labels had to be used to restrict the number of lines displayed. BBjDE, however, loads the entire program in a scrollable, editable text box. It's easy to scroll up and down to find the portion of code that you're looking for, so navigation is a breeze. Once you find the line of interest, you can edit it on the fly directly from the control. Editing a line in the program's context is conducive to more correct code because you're always aware of the surrounding routines. You no longer have to edit your program line by line.

To further aid you in making your code more legible, it is now possible to format your code and you have the option to make it case-sensitive. To illustrate the point, consider the following snippets of code. Which of these is easier to read and therefore maintain?

The old style, which forced uppercase for all non-string literals and line numbers:

```
       0010 IF DEBUG$="true" THEN IF SALARY>100000 THEN
                 PRINT "You're rich" ELSE PRINT
        0010: "Keep trying" ELSE PRINT "not in debug mode"
```

The new style, taking advantage of mixed case, indention and a non-numbered, multiple-line IF:

```
IF debug$="true" THEN
       IF salary>100000 THEN
              print "You're rich"
       ELSE
              print "Keep trying"
       ENDIF
ELSE
       print "not in debug mode"
ENDIF
```

The new case sensitivity and formatting options support and encourage better coding standards and more legible code. It is now easier than ever to figure out what a particular piece of unfamiliar code is supposed to do. After all, mixed-case variable names such as **'DayOfTheWeek'** are more readily recognized and understood than the uppercase **'DAYOFTHEWEEK'**.

Top Five Things You'll Love About BBjDE:

5. Always knowing what line is executing next.

4. Tracking your variables in the watch window.

3. No more ESCAPE commands.

2. Seeing your entire program in a scrollable, editable text box.

1. No more line numbers!

BBjDE also facilitates your debugging efforts by giving you a more complete picture of the state of your program. Instead of having to print the TCB(4) or dot-stepping to discover the current line number, BBjDE highlights the current line of code so you always know what will be executed next. The highlight will track the current line as your program executes, and BBjDE scrolls the code window to ensure that it is always visible.

BBjDE also furnishes a watch window so that you can keep an eye on your variables. The watch window maintains a list of variables that you're tracking, constantly updating their values during a program's execution to reflect their current state. Additionally, it is possible to modify a variable's contents from the watch window at run time, instantly changing its value. Troubleshooting with the watch window is more convenient than inserting ESCAPE commands and printing out a variable's value.

In fact, ESCAPE commands are now a thing of the past, rendered obsolete by BBjDE's breakpoints. By flagging a line as a breakpoint, you can effectively control the flow of your program. ESCAPE commands used to have to be hard-coded in your application during debugging sessions, potentially dangerous if you neglected to remove them later! Breakpoints, however, don't modify your code at all and can be quickly added or removed. Proper use of these breakpoints allows you to expertly control the execution of your program, skipping over large portions of code until you reach the line of interest.

BBj will move the BBx developer into the modern programming environment to which developers using younger languages have grown accustomed. BBjDE gives the BBx programmer a rich coding and debugging environment with new features that will work to collectively improve the programmer's efficiency and productivity and will eliminate the need for external tools in the development process.

Nick Decker first learned BASIC 15 years ago and has since programmed in BBx®, Visual Basic, Fortran, assembly language, C, Java™ and Perl, which gives him a great perspective for demonstrating the advances in the BBj™ development environment. He joined BASIS eight years ago as a Technical Support analyst and is now BASIS Engineering Supervisor.