EXTENDED BASIC GAME DEVELOPERS PACKAGE

(12/21/14)

For all its strengths, Extended BASIC has two major weaknesses. It cannot utilize the full power of the 9918A video processor, and programs generally run slowly. This package contains two applications that work together to address these shortcomings, making it possible to produce arcade quality games with XB.

XB256

XB256 is a collection of assembly language subroutines for Extended BASIC that unlock the graphics and sound capabilities of the TI-99/4A computer. No knowledge of assembly language is required to use XB256.

XB256 lets you select from two independent screens. Screen1 is the screen normally used by Extended BASIC and it is accessed as usual. Screen2 lets you define 256 characters, more than twice the 112 available in XB. Additionally, you can use up to 28 double sized sprites using the character definitions available to Screen1. You can toggle between the two screens as desired and preserve the graphics on each screen. When using Screen2 there are assembly subroutines that replace CHAR, CHARPAT, COLOR, and CHARSET. All other screen access in Screen2 is by the usual Extended BASIC statements such as PRINT, SPRITE, ACCEPT, etc.

Scrolling routines allow you to scroll screen characters left, right, up, or down. You can specify a window area for scrolling and leave the rest of the screen unchanged. Other routines let you scroll smoothly one pixel at a time to the left, right, up or down. Plus there is a text crawl that gives an effect similar to the STAR WARS title screen.

There are subroutines that let you hilight text, set the sprite early clock, print in any direction on the screen using all 32 columns, read from or write to the VDP ram, write compressed strings or sound tables to VDP ram, and play a sound list. A disk catalog subroutine has been added.

A utility (COMPRESS) is included that lets you save selected areas of VDP memory as compressed strings that can be merged with your program. This lets you save character definitions, sound tables, screen images, etc. in a more compact form that can be loaded virtually instantaneously.

There are two utilities (SLCOMPILER and SLCONVERT) that convert the CALL SOUNDs in an XB program to a sound table that contains music and sound effects in a form that can be loaded directly into VDP memory. This is much more compact, and your XB program can do other tasks while a sound list plays. After a sound table is loaded into VDP RAM you can play any sound list in it using CALL LINK(“PLAY”,address) When the sound list has started it will play automatically while your XB program does other things. Also, there is a second player that can play a different sound list simultaneously with the first. This way you can have background music playing and add sound effects without interrupting the background music.

If you are developing a program using XB256 that you intend to compile for increased speed, then you should be sure to keep in mind the limitations of the compiler. Refer to the compiler manual to find out more about these so you don't get stuck having to rewrite code.

COMPILER v2.56C

COMPILER v2.56C lets you take advantage of the simple program development offered by Extended BASIC, then make an end run around the speed limitations. The goal was to implement Extended BASIC as fully as possible within the time limits of the programmer and the memory limits of the machine. There *are* limitations and you may need to adjust your programming style a bit, but in general, all the major features of XB are supported. You can concentrate on writing and testing a program in the Extended BASIC environment. When the program has been perfected it can then be compiled into an equivalent code that will run at a speed approaching assembly language. The average Extended BASIC program will run about 20 times faster after being compiled, and certain operations can run up to 70 times faster.

The compiler has been expanded to include all the XB256 subroutines. Compiling a program that uses XB256 is no different than compiling a conventional XB program. Write the program in XB, test it until it is perfected, then use the compiler to get a huge performance increase. The compiler still has the same limitations regarding named subprograms, use of integer arithmetic, no trig functions, no nested arrays, no disk access, etc. But if you work within its limitations you can create arcade quality games.

If you are using XB256 to develop a program and then want to compile the program or run a compiled program, you should first return to the master title screen or type CALL LINK("OFF").

Each folder contains the documents in PDF format. The actual programs are contained in a disk image that can be copy/pasted into Win994a and an archived file that can be copied to Classic99 or transferred to a real TI99/4a. Fred Kaal's TI99Dir provides an easy way to do this.

Both XB256 and COMPILER256 can be run on a real TI99 with nothing more than XB, 32K and a disk drive. Although they are designed to complement each other, both these are stand alone utilities. Programs developed using XB256 do not have to be compiled. If the execution time is adequate they will run fine in XB, and you would have access to floating point math, disk access, etc. Similarly, XB programs do not have to use XB256; they can still be compiled for the increased speed.

There is one possible problem with using the compiler on a real TI99. In the past, the DV80 files that comprise the runtime routines would give an error when loading them. If this happens, you can load them using the E/A editor, which gives an error message after loading them. Then save them back to disk. Now they can be loaded without error. Do this to all six of the RUNTIME files.