Appendix 2: Speed and Space Hints

For some purposes it makes little difference how fast a program runs, while for others it is critical. The best way to make a program faster starts with a careful consideration of how it works and elimination of unnecessary steps. For example, a value computed within a FOR-loop which does not depend on the value of the FOR variable can often be computed once (before the loop) instead of many times (inside the loop).

In Extended XYBASIC integer arithmetic is faster than floating point arithmetic, and integer variable storage uses less memory space than floating point variable storage. When you know the value of a variable will always be an integer in the range -32767 to 32767, you conserve both speed and space by using an integer variable rather than a floating point variable. Similarly, execution of an integer FOR-loop is much faster than execution of the corresponding floating point FOR-loop. Since the subscripts of an array variable are always integers, you can save considerable conversion time by using integer variables rather than floating point variables in subscript formulas.

The execution speed of Extended XYBASIC programs which perform extensive string manipulation often may be improved by allocating additional string space. The extra string space lets XYBASIC require time-consuming garbage collection less frequently, and the resulting time savings can be dramatic. You can also save time by replacing a quoted string used repeatedly in string relations (for example, "a" in IF S$ >= "a" THEN...) with a reference to a string variable set to the given value outside the loop containing the relation (for example, A$ = "a" and IF S$ >= A$ THEN...).

Since XYBASIC is an interpreter, it looks up each variable used in a command every time the command is executed. The lookup procedure searches the symbol table starting with the most recently defined variable. XYBASIC spends a great deal of its execution time looking up variables, so rearranging programs to facilitate variable lookup can produce substantial speed improvements.

Several techniques let you speed up a program at the expense of using more memory space. Writing out a subroutine 'in-line' instead of using GOSUB makes the program longer, but saves the time used executing GOSUBs and RETURNs. Replacing an array variable with several simple variables often improves speed (since XYBASIC does not need to compute subscripts), although it usually results in an incomprehensible program.

Using XYBASIC's TRACE and BREAK commands for debugging also slows down program execution. Usually the execution speed when TRACEing is limited by the baud rate of the console; that is, XYBASIC waits for your console to print a line number before executing the next command. By repeatedly typing <control-O> to suspend and resume output, you can keep track of program execution while wasting less time waiting for console printing. Similarly, using the many powerful optional forms of BREAK instead of TRACE often lets your program run faster while you find bugs.

The ENABLE command slows down program execution a great deal, since the specified condition is checked before each command is executed. To improve program speed you should DISABLE interrupts whenever they are unnecessary, and naturally you should avoid ENABLEing interrupts which you do not need.