Source Code Optimization Rules of Thumb

☞ Learn to optimize intelligently. Concentrate on areas which are crucial or have high user visibility first.

☞ Avoid calling any trap which *only* sets simple structure values from the parameters passed into it.

☞ Allocate storage for variables tightly to decrease execution RAM requirements.

☞ Define routines to take character and Boolean parameters in two or four byte groupings.

☞ Don’t allow the source code to execute repeating code, either in a single routine or through the logic flow of the program, without a necessary reason.

☞ See if somebody has already written an optimized version of a function you need. Take sample timings and go with the faster version.

☞ Don’t overlock and unlock handles.

☞ Analyze the code flow and arrange code to avoid having to save and restore handle states.

☞ Use MoveHHi only when your are going to have a block locked for an extended period of time *or* you need to allocate a large block of memory and need the handle locked.

☞ Don’t call HNoPurge or HPurge needlessly.

☞ De-reference a handle into a pointer for repeated usage of the handle contents, to increase execution speed and decrease code size.

☞ Copy any instance variable used more than twice to a local variable for faster access.

❖ Be careful to maintain the current value if a called method may change or access the instance variable.

❖ If the local variable value changes, update the instance value when necessary.

☞ Bit shift for powers of 2 instead of using actual multiplication or division.

❖ Beware of bit shifting a negative number when using the Pascal BSR function.

☞ Add a value to itself in order to multiply by two.

☞ Create local variables to hold commonly computed subexpressions if an expression occurs three or more times for a simple expression or more than once for a complex expression.

☞ If a multiplication or division is being done more than once, store the result in a local variable.

☞ Use switch statements if possible over multiple if statements.

☞ Move the most common cases for a switch statement to the earliest evaluation positions.

☞ Use pointers where possible to work with arrays of data.

☞ Use Pascal strings to take advantage of the speed increase provided by a length byte.

☞ Let the compiler generate code to move data between structures.

☞ Use an optimized copy routine to move short strings around. Use BlockMove to move long string data around.

☞ Don’t auto-increment variables inline; Use a separate statement for that purpose.

☞ Use **do-while** loops to avoid a BRA instruction before a **for** or **while** loop.