SpeedDemon Profiler v1.2
Frequently Asked Questions

1. What kind of profiler is SpeedDemon Profiler?

SpeedDemon Profiler is a classic callsite wrapper or function header-footer profiler. It times the duration of function calls either at the call site to functions, or at the entry and exit points of functions, dependent on your compiler's instrumentation options.

2. What environment does SpeedDemon work in?

SpeedDemon Profiler currently works with C and C++ programs, built either with Microsoft Visual Studio 7 or greater, or Microsoft eMbedded Visual Studio 4.0 or greater.

Supported targets include:

Win32 (Windows 2000, Windows XP, Windows .NET Server 2003, etc)
- X86 (Pentium and higher)

Win64 (Windows XP x64 Edition, Windows .NET Server 2003, etc)
- x64 (AMD x64)

Windows Mobile 2003 (Windows CE 4.2, Pocket PC, and SmartPhone)
- PXA255 and PXA263 (PXA25x and PXA26x series)
- PXA270 (PXA27x series)

Windows Mobile 2005 (Windows CE 5.0, Pocket PC, and SmartPhone)
- PXA255 and PXA263 (PXA25x and PXA26x series)
- PXA270x (PXA27x series)

3. How do I use SpeedDemon Profiler in my project?

You build with debug symbols turned on (Program Database) select the instrumentation option of your compiler, and link all of your targets with the SpeedDemon shared library. For Win32 and Win64, the compiler options are “/GH /Gh”. For WinCE on other platforms, the compiler options is “/fastcap”. If “/fastcap” causes difficulties with the execution of your program, “/callcap” can be used as well, but will result in slightly less coverage area.

4. What kind of overhead do I incur with SpeedDemon Profiler?

You can expect a program that has 99% CPU utilization to slow down by about 4-5x, with about 75% to 80% of its time spent in the profiling code itself. Future versions of SpeedDemon will require less time as we optimize the hook functions.

5. What kind of data does SpeedDemon Profiler collect?

It generates statistics on a per-function basis, including Function Time (F Time), Function and Children Time (F+C Time), in Minimum, Average, Maximum, and Total variants.

6. How do I view the profiling results?

Use the program 'SPDReader' on the machine used to compile the program, or on a machine that has access to the debug symbols and source tree.

7. Can I filter on a subtree of the program?

Yes.

8. What about multithreaded environments?

SpeedDemon Profiler fully supports multithreaded environments, and keep track of timing on a per-thread basis.

9. What about dynamic library loading?

If you have a function in a library that you're profiling and you unload the library and load a different library in the same place, SpeedDemon Profiler support that completely. SpeedDemon Profiler doesn't support profiling dynamic libraries that are not compiled in an instrumented fashion. All modules to be timed must be compiled correctly and linked with SpeedDemon Profiler and have debug symbols available. This means, you can not time inside of system libraries with SpeedDemon Profiler.