

Energy-Efficient Software Criteria

Checklist for creating software applications that is energy-efficient. The checklist is operating system-agnostic except where otherwise noted.

Computational Efficiency

1 Loops

- Minimized the use of tight loops?
- Converted polling loops to being event driven?
- Using efficient polling loops (largest possible polling interval) when polling loops are necessary?
- Eliminated busy wait (spinning) loops, where possible? Might still be required for certain locking/synchronization tasks.

2 Performance Libraries/Extensions

- Utilizing instruction set extensions (such as SSE instructions, AVX, or newer) to accelerate computations?
- Utilizing Performance Libraries such as Intel® Performance Primitives or Intel® Math Kernel Library?
- Utilizing on-chip video encode/decode hardware acceleration using Intel® HD Graphics and the Intel® Media Software Development Kit (Intel® Media SDK)?

3 Algorithms

- Using high performance algorithms and data structures (gets the job done faster)?
- If requirements allow, can you select a less complex algorithm that is more power efficient?
- Minimized use of heavily recursive algorithms (these can be power inefficient)?

4 Compiler Optimization Options

- Is application optimized for speed?
- Is application optimized (profiled) for most common execution path?

5 Drivers

- Utilizing kernel/drivers that are idle power friendly?
- Identified drivers causing excessive interrupts?

6 Programming Language

- Using a programming language implementation and libraries that are idle power friendly?

Maximize Idle

10 Multithreading

- Taking advantage of multiple threads and cores to speed-up execution? Faster execution means more time to idle.
- Are threads balanced? Imbalanced threads may increase power consumption.

11 Reduce usage of High-Resolution Periodic Timers

- Using a timer rate appropriate for the application? Intervals < 15ms have low benefit on most systems and serve only to reduce idle times. Always make sure to disable periodic timer in case it is not in use.
- If application increases the timer tick rate when active, remember to reset to default timer value when idle? For Microsoft* Windows*, the default system timer resolution is 15.6ms.
- (Linux) Using timer APIs in application so that timers can be grouped (e.g. round_jiffies) or utilizing deferrable kernel timers?

Data Efficiency

- 20 Can data be pre-fetched from optical disk and buffered, allowing disk to idle?
- 21 Can data be pre-fetched from hard-disk and buffered, allowing disk to idle?
- 22 Avoiding frequent reads/writes to disk?
- 23 Can read/write requests be batched into one operation?
- 24 Using algorithms that minimize data movement, memory (cache versus RAM), and hierarchies that keep data close to the processor?
- 25 Identified processor resources shared between cores? Synchronize threads on different cores to work simultaneously and idle simultaneously.

Context-Aware (Power-Aware) Behavior

30 Handling Sleep Transitions Seamlessly

- Is application preventing/delaying system sleep or hibernate state transitions? Make sure to handle appropriate transition events.
- (Microsoft* Windows*) Are you preventing System Idle Timeouts only when necessary? Remember to reset execution state when task is complete (SetThreadExecutionState).

31 Respond/Adapt to System Power Events

- Responding to transition between battery and AC operation?
- Handling Low Battery event by saving work/state/checkpoint? Avoid duplicating work in case of standby.
- (Windows) Responding to PBT_APMSUSPEND within two seconds?
- Application adapting to user selected OS power policy?

32 Scale Behavior Based on Machine Power State

- Can resource usage be reduced when on battery power (such as avoiding background updates)?
- Is it possible switch to “low-power” algorithms on Low Battery?
- Depending on machine state, have you explored the option of letting the application inform the user to select a low(er) power profile for efficient execution?

33 Utilizing context awareness toolkits such as Intel® Mobile Platform SDK, Intel® Laptop Gaming TDK, or Intel® Web 2.0 TDK?

34 Can unused peripherals be shut off/disabled (e.g. Bluetooth*, 802.11) if no activity?

Testing for Energy-Efficiency

40 Profile system power during application runtime

- Understand the power impact of application at Idle and Running state
- Examine C-state behavior
- Examine P-state behavior
- Examine timer interrupts
- Examine interrupt statistics
- Examine disk and file access statistics

Tools

50 Intel® Power Checker (platform timer tick, idle power efficiency, power-awareness)	
51 Perfmon	Microsoft * Windows* XP, Windows Vista*, Windows 7
52 PwrTest/Windows Driver Kit	Microsoft Windows XP, Windows Vista, Windows 7
53 Windows Event Viewer (Timer tick change events, Microsoft-Windows-Kernel-Power\Diagnostic log)	Microsoft Windows XP, Windows Vista, Windows 7
54 Intel® PowerInformer	Microsoft Windows XP, Windows Vista, Windows 7
55 Windows ETW (performance monitoring framework)	Microsoft Windows XP, Windows Vista, Windows 7
56 Intel® Application Energy Toolkit	Microsoft Windows XP, Windows Vista, Windows 7, Linux, Apple* Mac OS*
57 PowerTOP	Linux
58 Battery Life Toolkit (BLTK)	Linux
59 Linux command-line tools: strace, blktrace, iostat, etc.	Linux

For details on the above topics and references to existing white papers and articles, please refer to the extended document, *Energy-Efficient Software Guidelines*. This checklist created by Intel as a general suggestion. If you have feedback, suggestions, or would like more information about power management go to <http://www.intel.com/software/power/>.

Intel, the Intel Logo, and MMX are trademarks of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2011, Intel Corporation. All rights reserved.