Power Efficient Software Coding

Asked 16 years, 3 months ago Modified 4 months ago Viewed 3k times

43

In a typical handheld/portable embedded system device Battery life is a major concern in design of H/W, S/W and the features the device can support. From the Software programming perspective, one is aware of MIPS, Memory(Data and Program) optimized code. I am aware of the H/W Deep sleep mode, Standby mode that are used to clock the hardware at lower Cycles or turn of the clock entirel to some unused circutis to save power, but i am looking for some ideas from that point of view:

Wherein my code is running and it needs to keep executing, given this how can I write the code "power" efficiently so as to consume minimum watts?

Are there any special programming constructs, data structures, control structures which i should look at to achieve minimum power consumption for a given functionality.

Are there any s/w high level design considerations which one should keep in mind at time of code structure design, or during low level design to make the code as power efficient(Least power consuming) as possible?

power-management embedded

Share

Improve this question

Follow

edited Aug 29, 2010 at 8:30

Pascal Thivent

570k ● 140 ● 1.1k ● 1.1k

asked Sep 15, 2008 at 5:06

goldenmean

18.9k ● 57 ● 157 ● 216

Agreed, it's no use to me but it's a really good question nonetheless :) – Teifion Sep 15, 2008 at 10:18

Why bother :-) From what I see most apps in handheld devices do not pay attention to battery life anymore :-(Luckily, operating systems still do – itj Sep 15, 2008 at 10:30

17 Answers

Sorted by: Highest score (default)

22

- Like `1800 INFORMATION` said, avoid polling; subscribe to events and wait for them to happen
- Update window content only when necessary - let the system decide when to redraw it
- When updating window content, ensure your code recreates as little of the invalid region as possible
- With quick code the CPU goes back to deep sleep mode faster and there's a better chance that such code stays in L1 cache
- Operate on small data at one time so data stays in caches as well

- Ensure that your application doesn't do any unnecessary action when in background
- Make your software not only power efficient, but also power aware - update graphics less often when on battery, disable animations, less hard drive thrashing

And read some other guidelines. ;)

Recently a series of posts called "Optimizing Software Applications for Power", started appearing on Intel Software Blogs. May be of some use for x86 developers.

Share Improve this answer Follow

edited Sep 4, 2014 at 21:26

answered Sep 15, 2008 at 5:42

macbirdie

16.2k ● 6 ● 49 ● 54

9

Zeroith, use a fully static machine that can stop when idle. You can't beat zero Hz.

First up, switch to a tickless operating system scheduler. Waking up every millisecend or so wastes power. If you can't, consider slowing the scheduler interrupt instead.

Secondly, ensure your idle thread is a power save, wait for next interrupt instruction. You can do this in the sort of under-regulated "userland" most small devices have.

Thirdly, if you have to poll or perform user confidence activities like updating the UI, sleep, do it, and get back to sleep.

Don't trust GUI frameworks that you haven't checked for "sleep and spin" kind of code. Especially the event timer you may be tempted to use for #2.

Block a thread on read instead of polling with select()/epoll()/ WaitForMultipleObjects(). Puts stress on the thread scheuler (and your brain) but the devices generally do okay. This ends up changing your high-level design a bit; it gets tidier!. A main loop that polls all the things you Might do ends up slow and wasteful on CPU, but does guarantee performance. (Guaranteed to be slow)

Cache results, lazily create things. Users expect the device to be slow so don't disappoint them. Less running is better. Run as little as you can get away with. Separate threads can be killed off when you stop needing them.

Try to get more memory than you need, then you can insert into more than one hashtable and save ever searching. This is a direct tradeoff if the memory is DRAM.

Look at a realtime-ier system than you think you might need. It saves time (sic) later. They cope better with threading too.

 answered Sep 15, 2008 at 5:37

Follow

Tim Williscroft
3,735 ● 25 ● 37

5

Do not poll. Use events and other OS primitives to wait for notifiable occurrences. Polling ensures that the CPU will stay active and use more battery life.

Share Improve this answer

Follow

answered Sep 15, 2008 at 5:17

1800 INFORMATION
135k ● 30 ● 163 ● 242

5

From my work using smart phones, the best way I have found of preserving battery life is to ensure that everything you do not need for your program to function at that specific point is disabled.

For example, only switch Bluetooth on when you need it, similarly the phone capabilities, turn the screen brightness down when it isn't needed, turn the volume down, etc.

The power used by these functions will generally far outweigh the power used by your code.

Share Improve this answer

Follow

answered Sep 15, 2008 at 7:23

Garry Shutler
32.7k ● 13 ● 89 ● 120

I second this. If you're using an embedded microcontroller like a PIC, disable the peripherals you are not actively using, like the A/D converters or the serial port. – Zebra North Sep 30, 2008 at 13:03

3

To avoid polling is a good suggestion.

A microprocessor's power consumption is roughly proportional to its clock frequency, and to the square of its supply voltage. If you have the possibility to adjust these from software, that could save some power. Also, turning off the parts of the processor that you don't need (e.g. floating-point unit) may help, but this very much depends on your platform. In any case, you need a way to measure the actual power consumption of your processor, so that you can find out what works and what not. Just like speed optimizations, power optimizations need to be carefully profiled.

Share Improve this answer

Follow

answered Apr 21, 2009 at 19:37

geschema

2,494 ● 4 ● 30 ● 48

1

Consider using the network interfaces the least you can. You might want to gather information and send it out in bursts instead of constantly send it.

Share Improve this answer

Follow

answered Sep 15, 2008 at 5:29

Omer van Kloeten
12k • 9 • 44 • 54

1

Look at what your compiler generates, particularly for hot areas of code.

Share Improve this answer

Follow

answered Sep 15, 2008 at 10:02

Ned
2,252 • 18 • 24

1

If you have low priority intermittent operations, don't use specific timers to wake up to deal with them, but deal with when processing other events.

Use logic to avoid stupid scenarios where your app might go to sleep for 10 ms and then have to wake up again for the next event. For the kind of platform mentioned it shouldn't matter if both events are processed at the same time. Having your own timer & callback mechanism **might** be appropriate for this kind of decision making. The trade off is in code complexity and maintenance vs. likely power savings.

Share Improve this answer

Follow

answered Sep 15, 2008 at 10:29

itj
1,165 • 1 • 11 • 16

Simply put, do as little as possible.

1

Share Improve this answer

Follow

answered Sep 30, 2008 at 13:19

Brian Knoblauch

21.3k ● 16 ● 61 ● 96

1

Well, to the extent that your code can execute entirely in the processor cache, you'll have less bus activity and save power. To the extent that your program is small enough to fit code+data entirely in the cache, you get that benefit "for free". OTOH, if your program is too big, and you can divide your programs into modules that are more or less independent of the other, you might get some power saving by dividing it into separate programs. (I suppose it's also possible to make a toolchain that spreas out related bundles of code and data into cache-sized chunks...)

I suppose that, theoretically, you can save some amount of unnecessary work by reducing the number of pointer dereferencing, and by refactoring your jumps so that the most likely jumps are taken first -- but that's not realistic to do as a programmer.

Transmeta had the idea of letting the machine do some instruction optimization on-the-fly to save power... But

that didn't seem to help enough... And look where that got them.

Share Improve this answer

Follow

answered Oct 21, 2008 at 22:59

Toybuilder

11.5k ● 5 ● 30 ● 34

1

Set unused memory or flash to 0xFF not 0x00. This is certainly true for flash and eeprom, not sure about s or d ram. For the proms there is an inversion so a 0 is stored as a 1 and takes more energy, a 1 is stored as a zero and takes less. This is why you read 0xFFs after erasing a block.

Share Improve this answer

Follow

answered May 18, 2009 at 1:54

old_timer

71.4k ● 9 ● 98 ● 174

> That seems like the micro optimizations of micro optimizations. – Earlz Dec 10, 2009 at 20:49

1

Rather timely this, article on Hackaday today about measuring power consumption of various commands: Hackaday: the-effect-of-code-on-power-consumption

Aside from that:
- Interrupts are your friends
- Polling / wait() aren't your friends
- Do as little as possible
- make your code as small/efficient as possible

- Turn off as many modules, pins, peripherals as possible in the micro
- Run as slowly as possible
- If the micro has settings for pin drive strengh, slew rate, etc. check them & configure them, the defaults are often full power / max speed.
- returning to the article above, go back and measure the power & see if you can drop it by altering things.

Share Improve this answer

Follow

answered Jun 14, 2012 at 15:31

John U

2,983 ● 3 ● 28 ● 40

0

also something that is not trivial to do is reduce precision of the mathematical operations, go for the smallest dataset available and if available by your development environment pack data and aggregate operations.

knuth books could give you all the variant of specific algorithms you need to save memory or cpu, or going with reduced precision minimizing the rounding errors

also, spent some time checking for all the embedded device api - for example most symbian phones could do audio encoding via a specialized hardware

Share Improve this answer

Follow

answered Sep 30, 2008 at 13:39

Lorenzo Boccaccia

6,121 ● 2 ● 22 ● 29

Do your work as quickly as possible, and then go to some idle state waiting for interrupts (or events) to happen. Try to make the code run out of cache with as little external memory traffic as possible.

Share Improve this answer

Follow

answered Sep 23, 2009 at 8:03

jakobengblom2

5,843 ● 2 ● 28 ● 34

On Linux, install powertop to see how often which piece of software wakes up the CPU. And follow the various tips that the powertop site links to, some of which are probably applicable to non-Linux, too.

http://www.lesswatts.org/projects/powertop/

Share Improve this answer

Follow

answered Dec 10, 2009 at 20:41

Peter Cordes

361k ● 49 ● 699 ● 958

Do you know what happened to the announced version 2.0 of PowerTop? The link is no longer available. – JJD Oct 8, 2011 at 23:31

1 really late reply that's probably easily solved with google, but v2.0 is released. 01.org/powertop – Peter Cordes May 20, 2012 at 3:21

Choose efficient algorithms that are quick and have small basic blocks and minimal memory accesses.

0

Understand the cache size and functional units of your processor.

Don't access memory. Don't use objects or garbage collection or any other high level constructs if they expands your working code or data set outside the available cache. If you know the cache size and associativity, lay out the entire working data set you will need in low power mode and fit it all into the dcache (forget some of the "proper" coding practices that scatter the data around in separate objects or data structures if that causes cache trashing). Same with all the subroutines. Put your working code set all in one module if necessary to stripe it all in the icache. If the processor has multiple levels of cache, try to fit in the lowest level of instruction or data cache possible. Don't use floating point unit or any other instructions that may power up any other optional functional units unless you can make a good case that use of these instructions significantly shortens the time that the CPU is out of sleep mode.

etc.

Share Improve this answer

Follow

answered Aug 29, 2010 at 8:50

hotpaw2

70.6k ● 15 ● 92 ● 155

Don't poll, sleep

0

Avoid using power hungry areas of the chip when possible. For example multipliers are power hungry, if you can shift and add you can save some Joules (as long as you don't do so much shifting and adding that actually the multiplier is a win!)

If you are really serious,I get a power-aware debugger, which can correlate power usage with your source code. Like this

Share Improve this answer

Follow

answered Jun 15, 2012 at 12:47

Martin Thompson

16.8k ● 1 ● 39 ● 58

Highly active question. Earn 10 reputation (not counting the association bonus) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.