Energy Metering for Free: Augmenting Switching Regulators for Real-Time Monitoring

Prabal Dutta[†], Mark Feldmeier[‡], Jay Taneja[†], Joseph Paradiso[‡], and David Culler[†]
{prabal,taneja,culler}@cs.berkeley.edu {carboxyl,joep}@mit.edu

[†]Computer Science Division
University of California, Berkeley
Berkeley, CA 94720

[‡]The Media Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139

1. Contact Person: Prabal Dutta, +1 510 495 4932 (tel), +1 510 642 5775 (fax), prabal@cs.berkeley.edu.

2. Designated Presenter: Prabal Dutta or Jay Taneja, TBD

3. An earlier version of this work appeared at IPSN'08 [1], which established the originality and distinguishing features of the design. Since that time, the iCount design has been integrated into a number of new hardware and software systems, which we would demonstate at ISLPED'08. Approval to submit this work to the ISLPED Design Contest was obtained from Yu Cao on May 22, 2008.

4. "All appropriate organizational approvals for the publication of this paper have obtained. If accepted the author(s) will select a designated speaker to present the paper at the Symposium." /s/ Prabal K. Dutta

Abstract

We present iCount, a new energy meter design. For many systems that have a built-in switching regulator, adding a single wire between the regulator and the microcontroller enables real-time energy metering. iCount measures energy usage by counting the switching cycles of the regulator. We show that switching frequency changes nearly linearly with load current for a variety of regulators.

1 The iCount Design

Many battery-operated devices use a switching regulator, as shown in Figure 1. Such regulators provide a constant output voltage and high conversion efficiency across a range of input voltages and load currents. Although a variety of regulator topologies (boost, buck, buck-boost), control modes (current-mode, voltage-mode) and modulation schemes (pulse-frequency modulated, pulse-width modulated) exist, we focus on boost regulators that employ current-mode control using pulse frequency modulation. Such regulators allow single-cell operation, can supply high currents, and draw ultra-low quiescent currents, making them ideal for low-power, battery-operated systems that exhibit a wide dynamic range in power draws. Unless otherwise noted, we use the terms *switcher* and *regulator* interchangeably in the remainder of this paper to describe PFM regulators.

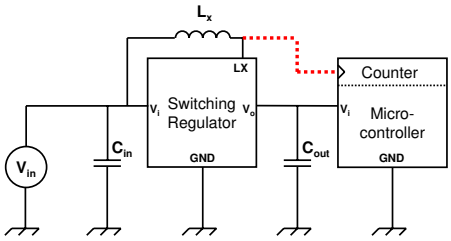

Figure 1. A typical system with switching regulator-based power supply, including the regulator, input capacitor, output capacitor, inductor, and load. Adding a single wire (the dashed line) between the regulator and a microcontroller counter input enables real-time energy metering.

A switcher goes through three stages during a switching *cycle*, as Figure 2 shows. Each cycle delivers $\frac{1}{2}Li^2$ J, where i is the peak inductor current (*i.e.* the max value of I_{LX}). A cycle begins when the switcher senses that the output has fallen below the regulation threshold. During the first stage of a cycle, the switch-side of the inductor (LX) is connected to ground. The resulting potential difference across the inductor gives rise to a steadily increasing current. When this current reaches a limit (or some maximum on-time has passed), the switch-side of the inductor is disconnected from ground. During the second stage of the cycle, the switch-side of the inductor is connected to the switcher output, which discharges the inductor energy into the output capacitor. When the inductor current ramps to zero, the discharge stage is complete. Sometimes, the inductor and output capacitor form a resonant circuit and this stage ends with ringing oscillations. During the third stage, the switcher maintains a quiescent state while the load draws current from the output capacitor. A cycle repeats when the output falls below the regulation threshold.

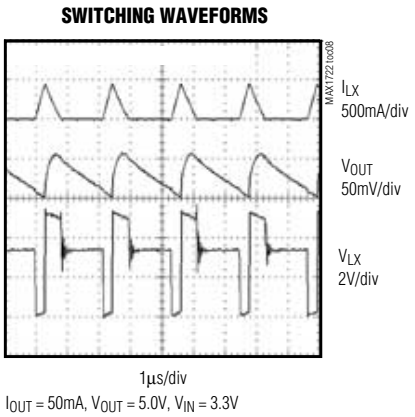

Figure 2. The switching waveform of the Maxim MAX1724 switcher. The inductor voltage, V_{LX}, alternates between the input voltage (3.3 V), ground (0 V), and output voltage (5.0 V). Source: Maxim [2].

The iCount design is motivated by the simple observation that many switchers exhibit a nearly linear relationship between switching frequency and load current over a wide dynamic range. Figure 3 shows how the switching frequency changes with load current for several different commercial switchers. In this figure, the bias, or no-load switching frequency, has been subtracted from each data point. Fortunately, such biases can be subtracted easily in software. Although the data appear linear over five decades, the data are linear only if the lines have unit slope since they are plotted on a log-log scale. We investigate the linearity in Section 3.

Since switchers provide a constant output voltage, the nearly linear relationship between switching frequency and load current implies a fixed amount of energy is delivered per cycle. Therefore, simply counting switching cycles approximates the total energy used over the counting interval, and dividing the number of counts by the counting interval gives the average power over that interval. However, because switchers do not expose their internal control logic and signals, one challenge is determining when the switcher actually cycles using only the signals that are observable externally. Our solution is to monitor the switch-side voltage, V_{LX}, of the inductor since it alternates between the input voltage, ground, and output voltage, as shown in Figure 2.

The basic iCount design follows directly from the data

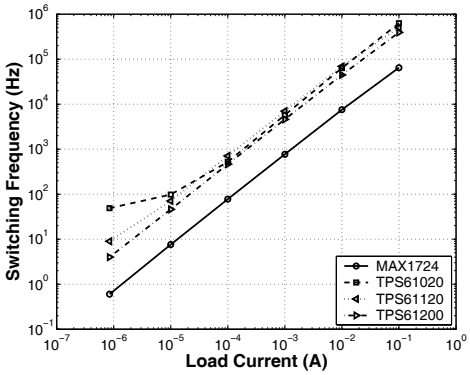

Figure 3. The relationship between load current and switching frequency for several switchers, after bias compensation. Some switchers are more linear than others (and sometimes over a wider dynamic range). Plotted on a log-log scale.

in Figures 2 and 3. Counting the rising edges of the V_{LX} signal is all that is required to accumulate energy usage. Since many battery-operated systems include a microcontroller, and since most microcontrollers support counters that can be externally-clocked, simply adding a single wire between the switcher and a the microcontroller's counter enables real-time energy metering.

2 Implementation

To evaluate the feasibility and performance of our design, we implemented iCount using off-the-shelf hardware and a small amount of driver software.

2.1 Hardware

Our implementation uses the Maxim MAX1724 [2] as the switcher and the Moteiv Tmote [3] sensor platform's MSP430 as the microcontroller. We chose the Maxim MAX1724 because of its low bias and ultra-low quiescent current. The majority of our experiments are based on the Maxim MAX1724 evaluation kit. We used a 3.3 V, fixed-output switching regulator (MAX1724EZK33); a pair of 10 μF, 16 V X7R ceramic capacitors for Cin and Cout (TDK C3225X7R1C106MT); and a 10 μH inductor (Sumida CDR43-100MC).

iCount requires a dedicated hardware counter to accumulate the switcher's cycles. We use Timer A on the Tmote's MSP430F1611 [4] for this purpose, since it is normally unused. In addition to connecting the Tmote's power supply lines to the switcher's output and ground, we added a single wire between the switch-side of the inductor and the Tmote's port U2.7, as shown in Figure 4(a). We populated resistor R16 with a zero-ohm resistor on the Tmote to connect port U2.7 to the TAINCLK line, the external clock for the MSP430's Timer A subsystem. Using this approach, each

Figure 4. (a) An illustration of how simple iCount is to implement: a Maxim MAX1724 switcher is connected to a Moteiv Tmote. (b) A sensor node for environmental monitoring includes iCount support which allows expected energy usage to be compared with actual energy usage. (c) A testbed node for benchmarking applications includes iCount plus five decades of precision calibration circuitry.

```
interface iCount {

  // Basic interface
  command error_t reset();
  command error_t start();
  command error_t pause();
  command uint32_t read();

  // Cycle-to-cycle feedback
  command error_t enableCapture();
  command errot_t disableCapture();
  async event void captured(uint32_t timestamp);

  // Quanta-based feedback
  command error_t compare(uint32_t delta)
  async event void matched(uint32_t now);
}
```

Figure 5. The iCount TinyOS API.

cycle of the switcher increments the hardware counter automatically and does not require the microcontroller to execute any software. We also removed diode D22 which ensures that only the Tmote USB interface is powered by USB when plugged into a computer and the microcontroller, radio, flash, sensors, and LEDs can be powered from a different source – in this case the switcher.

2.2 Software

We implemented our driver software in TinyOS, the *de facto* sensor network operating system. Our driver exposes the application programming interface shown in Figure 5. The driver also handles hardware counter overflows, wraps the underlying 16-bit counter in software to increase its width to 32-bits (or more), and ensures that counter state is accessed and modified safely.

If available, iCount can use capture and compare registers attached to the dedicated hardware counter to provide additional functionality. Since a capture records the value of the timebase whenever a rising edge corresponding to a regulator cycle occurs, it can be used to measure the interval between two successive cycles, and hence the near-instantaneous current of the hardware. A compare register, in contrast, allows an interrupt to be generated when a certain quanta of energy has been used. Although the MSP430 supports these

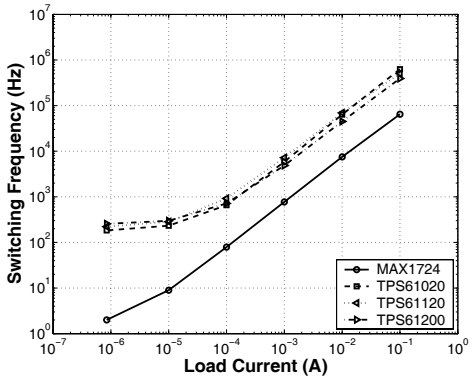

Figure 6. The relationship between load current and switching frequency without bias compensation shows significant non-linearity after three to four decades. Plotted on a log-log scale.

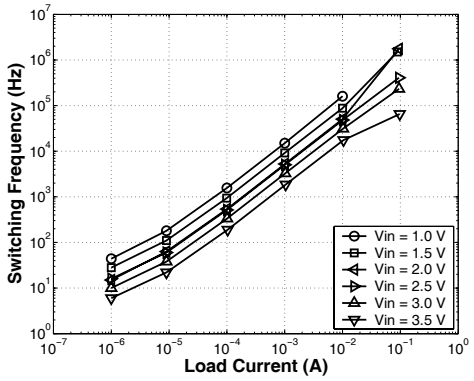

Figure 7. The relationship between input voltage and switching frequency. An order of magnitude variation in switching frequency occurs as input voltage goes from 1.0 V to 3.5 V. Plotted on a log-log scale.

features, the capture and compare functionality is currently unimplemented.

Switcher	Bias	Std Dev.	Min	Max
MAX1724	1.4 Hz	0.5 Hz	1 Hz	2 Hz
TPS61020	135 Hz	1 Hz	132 Hz	138 Hz
TPS61120	213 Hz	15 Hz	187 Hz	228 Hz
TPS61200	252 Hz	1 Hz	252 Hz	254 Hz

Table 1. The no-load minimum, average and maximum bias, and its standard deviation. At least 30 samples were taken, each with a gate period of 1 second.

2.3 Calibration

Viewing iCount as an instrument, calibration is necessary to establish the relationship between the input current and output frequency. Calibration requires measuring the bias and fitting a line to a set of current-frequency measurement pairs to minimize error.

Bias, or a switcher's no-load switching frequency, must be compensated. Removing the bias is necessary to ensure high accuracy measurements at small load currents. The bias is determined by disconnecting all loads from the switcher and measuring the output frequency at the switch-side of the inductor. An Agilent Technologies 53132 Universal Counter was used to measure the bias frequency using a gate period of 1 second. Figure 6 shows the effect of bias on the switching frequency vs. load current. The no-load bias for these switchers is shown in Table 1.

The relationship between input voltage and switching frequency is shown in Figure 7. An order of magnitude variation occurs as the switcher's input voltage is swept from 1.0 V to 3.5 V. This voltage-dependent relationship can be used to adjust the meter sensitivity at runtime as battery voltage slowly changes.

Linearity describes how well a line approximates the relationship between switching frequency and load current. A linear relationship is important for at least two reasons. First, iCount will be used in systems that exhibit a wide dynamic range in power draws, so linearity across this range is needed to faithfully capture the energy usage across the operating spectrum. Second, since iCount accumulates cycles, small non-linearities can result in large errors over time. These errors are bounded by the maximum non-linearity, so this figure is important.

To calibrate our implementation, we loaded the switcher in round-robin fashion using six resistors, measured the switching frequency for each value of the load resistance, and fit a line to the data using linear regression. We used the MAX1724 switcher with input voltage of 3.0 V and output voltage of 3.3 V. The six resistors had values of 33 Ω, 330 Ω, 3.3 kΩ, 33 kΩ, 330 kΩ, and 3.9 MΩ, which provided load currents ranging from about 1 μA to 100 mA. To weight these measurements equally during regression, we first took the logarithm of the current and frequency values, performed the linear regression, and then applied the exponential to complete the process.

3 Performance Metrics

In this section, we evaluate the performance of our iCount implementation using a set of microbenchmarks. These microbenchmarks assess the accuracy, resolution, overhead, read latency, responsiveness, precision, and stability of our implementation.

3.1 Accuracy

Figure 8 shows the accuracy of our implementation using the calibration coefficients previously described. We subtract

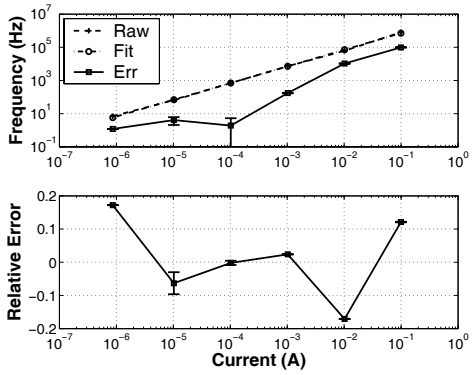

Figure 8. The accuracy and relative error. The relative error is less than $\pm 20\%$ over five decades of current draw. Plotted on log-log and lin-log scales, respectively.

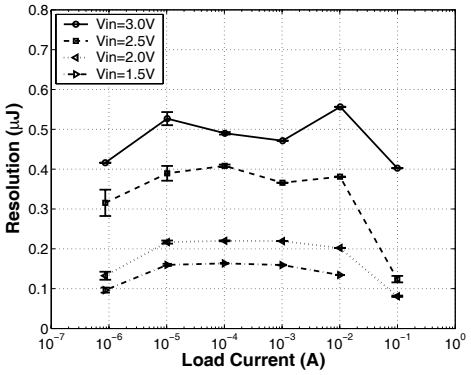

Figure 9. The resolution varies with input voltage and load current and exhibits high non-linearities at extreme load currents. Plotted on a lin-log scale.

the true measurement from the line of best fit to determine the absolute estimation error. To determine the relative error, we divide the absolute error by the true value. The relative error is less than $\pm 20\%$ across five decades of current draw. If we consider the typical operating range of a mote, from 5 μA to 50 mA, the relative error falls within $\pm 15\%$ range.

3.2 Resolution

Resolution refers to the smallest quanta of energy that the system can measure. To determine our implementation's resolution, we loaded the switcher in a round-robin fashion using six resistors, measured the switching frequency for each value of load resistance, and repeated across a total of four different voltages. The six resistors had values of 33 Ω, 330 Ω, 3.3 kΩ, 33 kΩ, 330 kΩ, and 3.9 MΩ, which provided load currents ranging from about 1 μA to 100 mA. We used an Agilent Technologies 53132 Universal Counter to measure the output frequency for each value of load resistance over a 1 second gate period. A Keithley 2400 sourcemeter, configured to use 4-wire sense mode to eliminate resistive losses along the power supply lines, provided 1.5 V, 2.0 V, 2.5 V, and 3.0 V. We estimated the energy delivered during each trial by squaring the output voltage, dividing by the load resistance, and multiplying the gate time:

$$E_{trial} = \frac{V_{out}^2}{R_{load}} T_{gate} \quad (1)$$

To determine the resolution (in μJ) or sensitivity (in μJ/cycle), we divided E_{trial} by the number of cycles, minus the no-load bias, during that trial. Figure 9 shows that the resolution of the system ranges from about 0.1μJ to 0.5μJ, depending primarily on input voltage and somewhat on load current. Note the significant reduction in resolution at high load currents. The root cause has not been determined but may be due to resistive losses in the switcher's current limiting circuitry, inductor saturation, or other losses associated with the damping circuitry. Since systems usually have a safety factor in their design, and do not operate close to or at the rated limits of the power supply, these non-linearities may not affect actual measurement performance but it is important to be aware of their existence.

The meter resolution depends on the value of the switching inductor. Each cycle of the switcher delivers $\frac{1}{2}Li^2$ J, where L is the inductance and i is the peak inductor current. Our experiments are based on a 10 μH inductor but note that a smaller inductor will provide a higher resolution, and vice versa. Also, note that manufacturing variations of $\pm 10\%$ are common, suggesting that individualized calibration is useful.

3.3 Overhead

There are three primary sources of overhead in the iCount design. First, the microcontroller-based hardware counter contributes to a fixed increase in the power draw. Second, the act of counting itself contributes a frequency-dependent increase in the power draw of the counter hardware due to gates changing state at increasing rates and the resulting charge/discharge cycles of the gate capacitance. Third, the microcontroller must service counter overflows periodically, with an average period equal to the average frequency divided by the maximum value of the hardware counter.

To determine the fixed overhead due to enabling the counter hardware, we programmed our mote with the TinyOS `Null` application, which forces all of the hardware components into their lowest power states. We powered the system using a Keithley 2400 sourcemeter configured to use 4-wire sense mode, to detect and eliminate any resistive losses in the power supply lines. Using a Fluke 189 digital multimeter, we measured the `Null` app current draw at 8.87 μA (using the DC-μA range setting) and 0.006 mA (on the DC-mA range setting), both averaged over one minute. This experiment established the baseline current draw of the mote.

We then added the iCount driver logic to the `Null` application, added a wire between the switch-side of the inductor and the counter clock input. Repeating the mea-

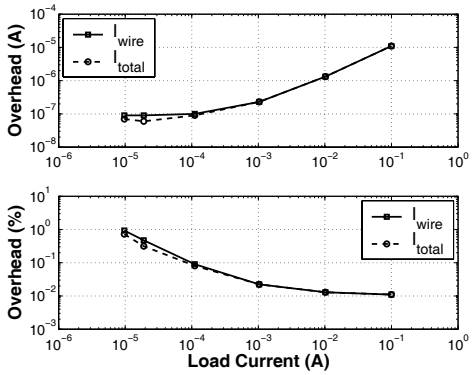

Figure 10. The overhead ranges from 0.01% for a 100 mA load to 1% for a 10 μA load, with a 3.0 V input. Plotted on a log-log scale.

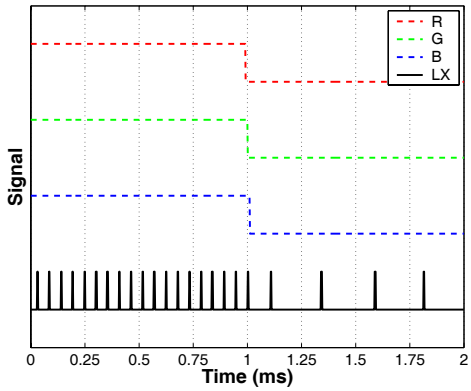

Figure 11. The system responds quickly to changes in the load current. The system adjusts in less than 125 μs to a load current decrease from about 10 mA to 2 mA.

surements, we found the current draw to be *the same* 8.87 μA as before via the microcontroller's power supply lines. But, we also found that a 90 nA current was being supplied by the switcher via the wire we added. This shows that the fixed overhead in our implementation is about 1% (90 nA/8.87 μA) when the system is in its lowest power state.

We also found that in some cases, the current draw via the power supply lines actually decreased, suggesting that some power was supplied via the counter wire. Figure 10 shows the overhead current across four decades of load current. The total current, I_{total}, is slightly lower than the current passing through the wire connecting the switcher and microcontroller, I_{wire}, because the total includes the decrease in current draw via the microcontroller power supply lines.

The iCount counter is implemented using the MSP430 microcontroller's 16-bit Timer A. Each time this hardware counter overflows, the microcontroller must increment a larger-width software counter by the maximum value of the hardware counter. Since this process requires writing the value of a variable in interrupt context, the write should be protected with an atomic section. All interrupts are turned off during the atomic section but since the hardware counter is clocked asynchronously, no cycles are missed during execution of the interrupt handler.

Since an overflow occurs after 65,536 cycles, and each cycle delivers approximately 0.5 μJ, the counter overflows once per 32,768 μJ of energy consumed. A typical mote draws about 25 mA at 3 V, when active, and would therefore draw about 750 μW running at a 1% duty cycle. At this 1% workload, the counter would overflow every 43 seconds (32,768 μJ/750 μW). In contrast, when fully active at a 100% duty cycle, the counter would overflow less than three times per second, making the overhead of handling counter overflows negligible.

3.4 Read Latency

Application software reads the hardware counter by calling `iCount.read`. This function, executed in an atomic section, disables the hardware counter, takes a snapshot, reenables the hardware counter, performs a 32-bit addition, and returns. The total time required to invoke this function, from call to return, is 15 μs. With this read latency, the counter could be read at over 66 kHz. We determined the read latency by setting an I/O pin on the microcontroller immediately before the call, setting a second I/O pin immediately after the call, and taking their difference. The overhead of setting an I/O pin was measured by setting two pins in succession, measuring their difference (1 μs), and subtracting this value from the prior uncalibrated measurement (16 μs).

3.5 Responsiveness

Responsiveness is a measure of how quickly the switching frequency settles when the load changes. To illustrate the responsiveness of the system, we instrumented the TinyOS `Blink` application and monitored the state of the red (R), green (G), and blue (B) LEDs as well as the cycles of the switcher. The greatest change in the power draw of this application occurs when the three LEDs are all turned off simultaneously. Figure 11 shows the change in the cycle rate of the switcher for a two millsecond period centered at the point when the LEDs are turned off. We see that the switching frequency adjusts to the new rate in less than 125 μs as the current draw falls from approximately 10 mA to 2 mA.

3.6 Precision

Precision characterizes the degree of mutual agreement among a series of individual measurements, or the ability to produce the same result given the same input conditions. To measure the precision of our implementation, we loaded the switcher with a fixed resistor and captured the inductor switching waveform over a two second window. This waveform was post-processed to extract the cycle-to-cycle switching period, and its reciprocal, the instantaneous frequency.

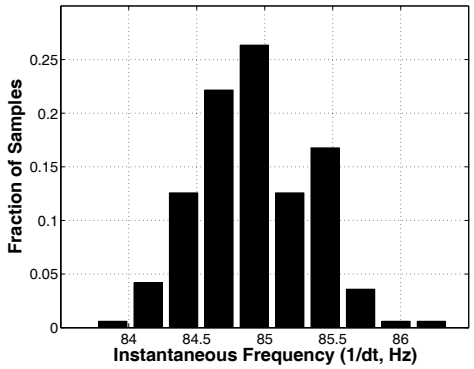

Figure 12. Measurement precision is better than $\pm$1.5%. The distribution of instantaneous (cycle-by-cycle) frequency measurements over a two second window.

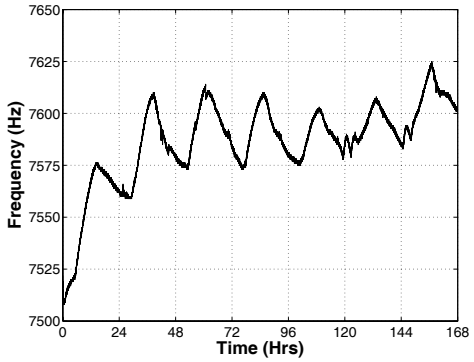

Figure 13. The system is stable. The range in variation is $\pm$1% of the mean over a one week period.

Figure 12 shows the distribution of these frequency values ($N = 167$). The variations in the values are small, within $\pm$1.5%, so a small number of samples provides an accurate estimate of the near-instantaneous current.

3.7 Stability

While precision characterizes the degree of mutual agreement among a series of individual measurements over the short term, stability refers to the agreement of these readings over a much longer term. To evaluate the stability of our implementation, we loaded iCount with a fixed resistor and used a mote-based application to track the switching frequency over a one week period. The measurement setup was placed near a window and experienced daily temperature variations of more than 10° C. Figure 13 illustrates how the switching frequency changed over the course of the week. The range in variation falls within $\pm$1% of the mean.

The data show both daily fluctuations in frequency as well as a one-time "warmup" period. The daily variations may be due to temperature- or humidity-dependent changes in the inductance or temperature-dependent changes in the measurement system (an uncalibrated 32 kHz crystal oscillator was used). The one-time warmup period may be an artifact of the experimental setup: the iCount system was located near a laptop computer that vented warm air.

It might seem odd that our implementation exhibits a stability of $\pm$1% over one week while only exhibiting a precision of $\pm$1.5% over a two second window. One plausible explanation is that for the precision measurements, we presented the instantaneous (or cycle-by-cycle) frequency but for the stability measurements, we averaged over a one second period.

4 Conclusion

Traditional instrument-based power measurements, which are useful for design-time laboratory testing, are impractical for everyday *in situ* use due to the cost of instruments, their physical size, and their poor system integration. Of course, dedicated power metering hardware can enable runtime adaptation, but this approach results in increased hardware costs and power draws. In this work, we demonstrate that energy metering is possible for free by simply counting the cycles of a switching regulator.

Our particular implementation exhibits a maximum error of less than $\pm$20% over five decades of current draw, a resolution exceeding 1 μJ, a read latency of 15 μs, and a power overhead that ranges from 1% when the node is in standby to 0.01% when the node is active, for a typical workload. The basic iCount design requires only a pulse frequency modulated switching regulator and a microcontroller with an externally-clocked counter. iCount enables practical, *in situ* energy metering for many systems which may not lend themselves to traditional instrumentation techniques. We show that with the addition of a single wire, iCount enables a device to introspect its own energy usage with virtually no cost or energy overhead.

References

[1] P. Dutta, M. Feldmeier, J. Paradiso, and D. Culler. Energy metering for free: Augmenting switching regulators for real-time monitoring. In *IPSN '08: Proceedings of the 2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*, pages 283–294, 2008.

[2] Maxim Integrated Products. 1.5uA IQ, Step-Up DC-DC Converters in Thin SOT23-5. http://datasheets.maxim-ic.com/en/ds/MAX1722-MAX1724.pdf, July 2001.

[3] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power research. In *Proceedings of the 4th International Conference on Information Processing in Sensor Networks (IPSN '05)*, Apr. 2005.

[4] Texas Instruments. Mixed signal microcontroller (rev. e). http://www.ti.com/lit/gpn/msp430f1611, Aug. 2006.