Using BatterySmart™ to Interpret and Regulate Handheld Device Power Consumption

30 West Gude Dr, Suite 100
Rockville, MD 20850
240-558-2014
information@inhandelectronics.com
www.inhandelectronics.com

1 Introduction

BatterySmart is a series of hardware and software technologies that are incorporated on all of InHand's handheld device platforms, including Elf™ and Fingertip™. The main design goal for BatterySmart is to reduce power consumption to the lowest possible level, for a given handheld device application environment.

This white paper describes BatterySmart in greater detail, and explains the results of a variety of empirical tests of power consumption on the Elf and Fingertip platforms. In addition to demonstrating the low power capabilities of BatterySmart, the results are also quite instructive in providing you with a better understanding of the relative effects of various system components on power consumption.

2 Low Power Hardware Design

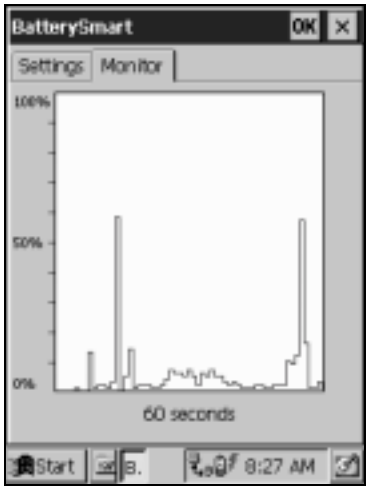
Designing high-performance, low-power handheld devices starts with choosing Intel's StrongARM® microprocessor. At 206MHz, it consumes only about 400mW, giving it an outstanding combination of high computational performance and low power consumption. However, to create a truly low-power StrongARM-based handheld device requires proper design of the power supplies, power interfaces, and peripheral interfaces.

Power supply design is an area where certain approaches can adversely impact power consumption. For example, some commercial StrongARM-based boards and reference platforms use simplistic techniques to convert and regulate DC battery voltage. These approaches result in inefficient power regulation, causing a potentially significant increase in board-level power consumption, as compared with the more optimal designs used on InHand's Elf and Fingertip platforms. This is one reason why some other StrongARM platforms have power consumption exceeding two (2) watts. This extra power on these other platforms is wasted in the form of heat. Heat is another potential problem source in handheld devices, since small, tight-fitting enclosures typically do not provide good thermal regulation.

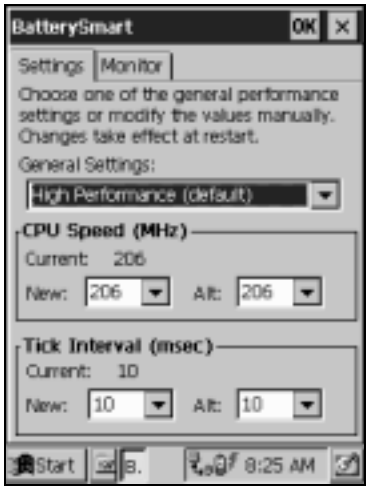
3 BatterySmart Software

The BatterySmart software takes advantage of the low-power designs of the Elf and Fingertip platforms. BatterySmart includes three components: a sophisticated CPU monitor; an interface for manipulating CPU performance parameters such as clock speed and operating system scheduler tick interval; and driver-level settings that minimize the power consumption of software drivers and peripherals.

The CPU monitor works at the OS kernel level and measures the amount of time that StrongARM spends in its Idle mode. Idle time measurements are a highly accurate reflection of the computational bandwidth that the CPU is using. These measurements are brought out to the developer and user through the BatterySmart control panel applet, as shown below.



By analyzing these measurements as compared with StrongARM's clock speed, you can determine if your handheld device can operate adequately at a lower clock speed. If so, BatterySmart allows you to manipulate StrongARM's clock speed, as shown in the control panel applet below.



In analyzing performance and propagating the actual CPU clock speed settings to the appropriate software drivers at startup, BatterySmart gives you the ability to adjust the CPU clock speed to the requirements of your application. As shown in the measurements in the following sections, this capability can result in power reductions of over 50%.

At the software driver level, each software driver is tested and qualified relative to its effect on power consumption within the handheld device. Thus, in addition to testing software drivers for performance and correctness, power issues are analyzed as well.

A good example of this is in the display software drivers. StrongARM's LCD controller includes a variety of interrupt sources that can be activated based on error conditions and notifications that occur during operation. These interrupt sources are often not relevant in applications, and can be ignored altogether by masking the appropriate interrupt bit in StrongARM's interrupt controller. However, since the registers within the LCD controller are used to enable/disable the interrupt source from even reaching the interrupt controller and since these registers are not guaranteed to come up in a predefined state at reset, it is possible that interrupt sources in the LCD controller will reach the interrupt controller.

If LCD controller and interrupt controller registers are not initialized properly, LCD interrupt sources will wake StrongARM from its Idle mode, even if they are masked. In this case, if an application is using the Idle mode to reduce power consumption when the application is not operating, power consumption might be reduced by up to 70% simply by proper setting of interrupt-related registers.

4 Power Consumption Experiments

To get a better understanding of the factors that effect power consumption, a series of experiments were performed on the Elf and Fingertip platforms. The results are summarized in the following sections and are provided in complete tabular form in the appendices.

4.1 Experimental Setup

Power consumption experiments were performed using both the Elf and Fingertip handheld device platforms. **These experiments measured not just the board's power consumption, but actual *real-world* handheld device power consumption, with a Hitachi 320x240 monochrome LCD display attached to each platform and the Windows CE operating system (version 3.0) running.** A voltmeter and ammeter were connected and utilized to measure the total voltage and current used by the board.

4.2 Measurements

A variety of operational scenarios were initiated and voltage and current measurements were used to determine total power consumption. In these scenarios, a variety of system parameters were manipulated, including CPU clock speed, operating system scheduler tick interval, CPU bandwidth utilization, peripheral access, and LCD backlight.

4.2.1 Effects of CPU Clock Speed and Bandwidth

The first experiments isolated CPU performance. A "bandwidth-eating" software application was created that allowed experimenting with various levels of StrongARM computational bandwidth usage, at multiple clock speeds. In this case, computational bandwidth was defined as the percentage of time that the CPU was actually performing a task and not in Idle mode. Bandwidth usage was verified and CPU clock speed adjustments were made via the BatterySmart control panel applet.

The chart below shows power consumption at various CPU computational bandwidths and at various clock speeds for both Elf and Fingertip (OS scheduler tick interval of 10ms).

CPU Speed (MHz)	CPU Bandwidth Utilization (%)	Power Consumption (mW) Elf	Power Consumption (mW) Fingertip
59	0	344.4	346.8
59	20	394.5	397.6
59	40	442.7	447.0
59	60	493.7	499.4
59	80	546.9	550.6
59	100	597.8	598.6
133	0	368.1	353.7
133	20	484.4	472.3
133	40	603.6	591.4
133	60	720.0	706.0
133	80	836.4	823.6
133	100	958.3	946.7
206	0	397.2	412.6
206	20	587.1	605.4
206	40	776.1	800.0
206	60	966.6	992.2
206	80	1147.4	1175.6
206	100	1337.3	1368.5

The power consumption is essentially comparable for both the Elf and Fingertip platforms. An interesting point is that the power consumption numbers are directly proportional to both bandwidth and CPU clock speed. In fact, the following equation can be used to provide a reasonable estimate of Elf's power consumption, as a function of CPU clock speed (*MHz*) and CPU computational bandwidth (*BW*):

$$MilliWatts = 326 + (0.322 * MHz) + (4.5 * MHz * BW)$$

In this equation, the first term represents the constant power that the system requires, independent of how fast the CPU is executing and what it is doing. This constant power is due to portions of the CPU, power supply, LCD, and other components on the board that are operational. The second term is generally related to those items directly affected by CPU speed, such as the LCD controller and memory interface. The third term measures power consumption based on computational performance and is proportional to both clock speed and bandwidth.

4.2.2 Effects of OS Scheduler Tick Interval

Multi-tasking operating systems such as Windows CE typically have a task scheduler that doles out time-slices to individual threads at a given priority level. In Windows CE, the handheld device developer typically adjusts the OS scheduler tick interval. Reducing the interval results in shorter time-slices and generally higher responsiveness to user input, at the expense of more thread-switching overhead.

In a scenario where no computational bandwidth of the CPU is being used, short time-slices are wasteful, since the tick interrupt wakes the CPU out of its Idle mode, only to have the OS determine that no threads are available to run and immediately return to Idle mode. As repeatedly waking the CPU out of Idle mode can have a negative impact on power consumption, the next

table shows measurements of the effect of varying the OS scheduler tick interval, at 206MHz, with 0% computational bandwidth.

OS Scheduler Tick Interval (ms)	Power Consumption (mW) Elf	Power Consumption (mW) Fingertip
5	398.3	408.3
10	397.0	406.3
20	398.4	408.3
25	397.9	408.0
50	397.8	408.0

Interestingly, the effect on power consumption of adjusting the OS scheduler tick interval is negligible. Thus, while there are reasons to adjust the timer tick interval, reducing power consumption is not one of them.

4.2.3 Effects of PCMCIA and CF Peripherals

Many applications require the use of PCMCIA or Compact Flash (CF) card peripherals. These cards can be quite high in power consumption, and in some cases, higher in power consumption than the rest of the handheld device's electronics. To quantify the effect of these peripherals, power consumption measurements were taken with several common cards. The table below shows these measurements.

Peripheral	Task	CPU Speed (MHz)	Power Consumption (mW) Elf	Power Consumption (mW) Fingertip
ATA-Flash	Plugged In	59	347.0	349.0
ATA-Flash	Write	59	719.4	573.5
ATA-Flash	Read	59	679.7	538.3
ATA-Flash	Plugged In	133	369.4	350.8
ATA-Flash	Write	133	900.6	793.9
ATA-Flash	Read	133	1019.1	849.1
ATA-Flash	Plugged In	206	399.3	411.3
ATA-Flash	Write	206	1032.0	922.7
ATA-Flash	Read	206	1313.4	1138.8
Hard Drive	Plugged In	206	751.3	N/A
Hard Drive	Write	206	2945.8	N/A
Hard Drive	Read	206	3228.8	N/A
IEEE 802.11b	Plugged In	206	1436.4	N/A
IEEE 802.11b	Power Saver	206	499.4	N/A

Many interesting results are shown in this table. First, it is encouraging that ATA-Flash cards do not draw significant power when plugged in and idle. The same cannot be said for the hard drive or the IEEE 802.11b wireless cards that were tested, as both draw significant power when idle. The wireless card at least has a power saving mode, which does reduce power consumption to a more manageable level, yet still allows the card to seek out remote wireless nodes.

The most interesting results though may be related to the power consumption when reading and writing ATA-Flash cards at different CPU clock speeds. As a general rule (not including the

Fingertip results at 59MHz), reading requires more instantaneous power consumption than writing. This may be due to a variety of factors, including that the write process to Flash requires longer intervals while the Flash is burned than reads do, and therefore may result in more time-periods of inactivity within a given interval.

In this case though, the goal is to complete a task, such as transferring a large file, and the real measure of note is energy consumption (power consumption multiplied by the amount of time to perform the task). For example, the following table shows energy consumption measurements for large file transfers, which use the power consumption measurements from the above table but which factor in the amount of time required to perform a specific task.

Peripheral	Task	CPU Speed (MHz)	Energy Consumption (Joules) Elf	Energy Consumption (Joules) Fingertip
ATA-Flash	Write	59	17.99	22.37
ATA-Flash	Read	59	33.99	50.06
ATA-Flash	Write	133	18.01	17.47
ATA-Flash	Read	133	24.46	21.23
ATA-Flash	Write	206	19.61	18.45
ATA-Flash	Read	206	22.33	20.50
Hard Drive	Write	206	41.24	N/A
Hard Drive	Read	206	54.89	N/A

The 133MHz and 206MHz energy measurements show significant correlation, indicating that while it may take longer to do a large file transfer at a slower CPU clock speed, the total energy consumption during the interval is similar to that at the higher clock speed. At 59MHz, timing issues begin to arise related to how the StrongARM CPU interfaces to the PCMCIA or CF peripheral. At these lower clock speeds, the settings for duration of peripheral accesses are more granular, giving you less ability to set access times at optimal levels. As a result, transferring large files can take longer, and can have an adverse affect on performance and power consumption.

4.2.4 Effects of Serial Port Access

Many handheld devices use serial ports to communicate with desktop PCs for data synchronization. The following table shows power consumption at 206MHz, when the Elf and Fingertip platforms are connected to a PC via Windows CE ActiveSync.

Task	Power Consumption (mW) Elf	Power Consumption (mW) Fingertip
ActiveSync Connected	431.8	448.6
Transfer From PC	446.3	454.8
Transfer To PC	440.4	454.9

Appendix A – Power Consumption Measurements for Fingertip

Task	CPU Speed (MHz)	OS Scheduler Tick Interval (ms)	Power Consumption (mW)
0% CPU Bandwidth	206	5	408.3
0% CPU Bandwidth	206	10	406.3
0% CPU Bandwidth	206	20	408.3
0% CPU Bandwidth	206	25	408.0
0% CPU Bandwidth	206	50	408.0
0% CPU Bandwidth (backlight on)	206	10	501.8
ATA-Flash Card Plugged In	59	10	349.0
Write ATA-Flash Card	59	10	573.5
Read ATA-Flash Card	59	10	538.3
ATA-Flash Card Plugged In	133	10	350.8
Write ATA-Flash Card	133	10	793.9
Read ATA-Flash Card	133	10	849.1
ATA-Flash Card Plugged In	206	10	411.3
Write ATA-Flash Card	206	10	922.7
Read ATA-Flash Card	206	10	1138.8
ActiveSync Connection	206	10	448.6
ActiveSync Transfer To PC	206	10	454.8
ActiveSync Transfer From PC	206	10	454.9
0% CPU Bandwidth	59	10	346.8
20% CPU Bandwidth	59	10	397.6
40% CPU Bandwidth	59	10	447.0
60% CPU Bandwidth	59	10	499.4
80% CPU Bandwidth	59	10	550.6
100% CPU Bandwidth	59	10	598.6
0% CPU Bandwidth	133	10	353.7
20% CPU Bandwidth	133	10	472.3
40% CPU Bandwidth	133	10	591.4
60% CPU Bandwidth	133	10	706.0
80% CPU Bandwidth	133	10	823.6
100% CPU Bandwidth	133	10	946.7
0% CPU Bandwidth	206	10	412.6
20% CPU Bandwidth	206	10	605.4
40% CPU Bandwidth	206	10	800.0
60% CPU Bandwidth	206	10	992.2
80% CPU Bandwidth	206	10	1175.6
100% CPU Bandwidth	206	10	1368.5

Appendix B – Power Consumption Measurements for Elf

Task	CPU Speed (MHz)	OS Scheduler Tick Interval (ms)	Power Consumption (mW)
0% CPU Bandwidth	206	5	398.3
0% CPU Bandwidth	206	10	397.0
0% CPU Bandwidth	206	20	398.4
0% CPU Bandwidth	206	25	397.9
0% CPU Bandwidth	206	50	397.8
0% CPU Bandwidth (backlight on)	206	10	492.1
ATA-Flash Card Plugged In	59	10	347.0
Write ATA-Flash Card	59	10	719.4
Read ATA-Flash Card	59	10	679.7
ATA-Flash Card Plugged In	133	10	369.4
Write ATA-Flash Card	133	10	900.6
Read ATA-Flash Card	133	10	1019.1
ATA-Flash Card Plugged In	206	10	399.3
Write ATA-Flash Card	206	10	1032.0
Read ATA-Flash Card	206	10	1313.4
Hard Drive Card Plugged In	206	10	751.3
Write Hard Drive Card	206	10	2945.8
Read Hard Drive Card	206	10	3228.8
IEEE 802.11b Card Plugged In	206	10	1436.4
IEEE 802.11b Card Power Mgmt.	206	10	499.4
ActiveSync Connection	206	10	431.8
ActiveSync Transfer To PC	206	10	440.4
ActiveSync Transfer From PC	206	10	446.3
0% CPU Bandwidth	59	10	344.4
20% CPU Bandwidth	59	10	394.5
40% CPU Bandwidth	59	10	442.7
60% CPU Bandwidth	59	10	493.7
80% CPU Bandwidth	59	10	546.9
100% CPU Bandwidth	59	10	597.8
0% CPU Bandwidth	133	10	368.1
20% CPU Bandwidth	133	10	484.4
40% CPU Bandwidth	133	10	603.6
60% CPU Bandwidth	133	10	720.0
80% CPU Bandwidth	133	10	836.4
100% CPU Bandwidth	133	10	958.3
0% CPU Bandwidth	206	10	397.2
20% CPU Bandwidth	206	10	587.1
40% CPU Bandwidth	206	10	776.1
60% CPU Bandwidth	206	10	966.6
80% CPU Bandwidth	206	10	1147.4
100% CPU Bandwidth	206	10	1337.3