PEN
Personal Electronic Notebook

Technical Description and Summary

University of Utah
Computer Engineering Program
2006

Jacob Shaha
Neal Tew
Clark Coulam

Introduction

Paper is the foundation of modern society. By itself, paper doesn’t seem like much of a building block, but when the whole of recorded history, research, and literature depends on paper to convey it from one generation to the next, then paper’s vital role in our lives becomes more apparent. Add in the countless trivial uses that everyone has for paper – from taking notes in class, to working out problems, to writing grocery lists – and it’s not hard to see why paper is available in every color, shape and size, and can be found in nearly every corner of the inhabited Earth.

Paper offers a wide variety of excellent features: it is energy efficient, cheap, compact, has a wide viewable angle, a simple interface, and no content restrictions or user constraints (other than the physical constraints of not breaking the paper). But paper does have a significant drawback: it piles. More specifically, as we use up paper, we often need to save what we’ve written on it, and that saved paper consumes more and more space.

PEN (Personal Electronic Notebook) is intended to replace the piles of paper that are so ubiquitous in modern life. PEN provides a cost-effective, organized, intuitive, “green” alternative to paper. It offers the advantage of an electronic first-copy, reducing the need to type notes after the fact or scan documents for electronic manipulation or distribution. It also sports an interface that is just as simple and intuitive as real paper. PEN is battery efficient, offering 20 hours of use between recharges, and has enough storage on unit for nearly 2000 pages. Since PEN can offload those pages to a larger electronic storage device, such as a home PC, PEN offers consumers the chance to remove every pile of paper in their lives and replace them with a single, simple alternative.

For PEN to meet the needs currently met by paper, it must be simple to use and resistant to failure. Paper’s simplicity and versatility are mirrored in PEN’s interface. Most importantly, PEN is designed from the ground-up to *never* “crash,” as a traditional computer might. Just as a sheet of paper never “locks up,” PEN’s hardware-level operating system is much more resilient than a notebook or laptop computer. The PEN philosophy is: Simpler and Sturdier.

Functionality

PEN presents users with a natural and simple interface. The viewing screen is divided into two portions: the *paper* and the *dashboard*. The Dashboard is the lower portion of the screen and is divided into buttons and a single readout; the paper is the upper, majority portion of the screen and can be drawn on with the stylus. Drawing to the paper is as simple as pressing down on the stylus. One end of the stylus will draw a thin, black line that follows the pen-point. The other end will remove existing black lines with a broad eraser brush.

The dashboard has several intuitive buttons. The **Bookmark** button marks the currently visible page; the effect of bookmarking a page will be discussed shortly. The **Background** button toggles the various backgrounds for the currently viewable page. The **Insert Blank** button inserts a new, blank page, sequentially in front of the current page. The **Insert Copy** button inserts a copy of the current page in front of the current page. Finally, the **Delete** button destroys the currently viewed page.

The **Next page** and **Previous page** buttons move to the next and previous sequential pages, respectively. The **Next Bookmark** and **Previous Bookmark** buttons move forward or backward in sequence until a bookmarked page is encountered; this makes dividing PEN's many pages into sections simple, providing an intuitive way to create organization in the notebook.

Technical Specs

PEN Hardware

PEN is centered around an Intel PXA255 ARM processor, embedded in a Gumstix 400xm motherboard with a Gumstix Breakout gs daughterboard. The motherboard/daughterboard environment give the processor access to an LCD display controller, a USB interface, 3 UART interfaces, multiple GPIO signals, and other accessories (those used will be specifically referred to as introduced). All these resources are pre-mapped into the processor's memory, allowing rapid DMA access. The PXA255 uses ARM assembly, for which many open-source assemblers and compilers are available. The current version of PEN's software was compiled from C and assembled into ARM assembly by the Arm ADT software package, freely available from the ARM corporation.

The CPU and all hardware provided on-board runs at 5 VDC; this includes the Flash memory and SDRAM.

The high-level hardware component relationship is illustrated in figure 1:

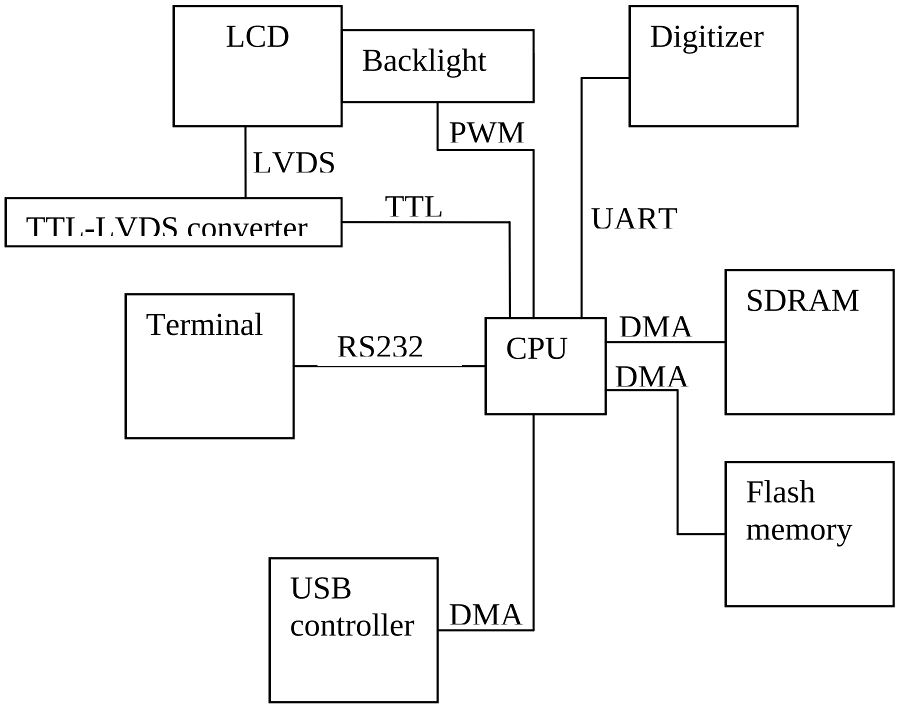

Figure 1: Hardware block diagram

The primary output mechanism is the display. The display actually consists of two separate hardware components: and LCD and a backlight. The backlight is simply a powered lamp that takes a pulse-width modulated signal as input for brightness control; the CPU generates this signal and it is directly fed to the lamp. There is also a single line used for reset, to reactivate the backlight after power interruption. The display takes an LVDS serial signal, a common LCD standard. However, the Gumstix LCD controller outputs a TTL parallel signal, a competing LCD standard. To bridge these components, PEN includes an EBKLVDS2 converter, which takes a TTL input and produces an LVDS output. The converter is powered by externally provided 3.3 VDC.

The backlight runs on DC voltage, from 7V to 15V (theoretical; we never actually "pushed" the upper voltage limit). This power is largely used to drive the screen inverter, which steps the DC voltage up to a high-voltage AC (>100 VAC) signal at about 56 kHz. The inverter's output drives the lamp used by the backlight. The display uses 3.3V logic to control the liquid crystal array based on LVDS input.

Because the TTL and LVDS standards are very specifically defined in generally available literature, they are for simplicity's sake excluded from this document.

The CPU has access to 64 megabytes of SDRAM for fast-access memory. The memory is "internal" to the CPU, meaning to explicit external memory access mechanisms are required. Similarly, the CPU has access to 16 megabytes of Flash memory for permanent (powerless) storage. Although the Flash is likewise "internal" to the CPU, is cannot be accessed in the same way as the SDRAM. Because Flash memory is, by design, block erasable only, the Flash memory is treated largely as an external component. The memory is “primed” by sending specific commands to its block-0 memory address; these commands set the memory to “read-mode” or else instruct the Flash to erase specific blocks and, if necessary, to write data to those recently erased blocks. In read-mode, the Flash behaves simply as an extension of internal memory, in that it can be randomly accessed via the familiar memory structure; for writing purposes, however, the Flash is treated as an external device, and data is exported through the established command protocols.

The Flash memory was manufactured by Intel, and the command structure for interacting with it is freely available online from Intel Corp. For specifics on PEN’s implementation, refer to the source code in the appendices of this document.

PEN’s primary input mechanism is a Wacom digitizer and stylus. The digitizer uses 3.3VDC to power its logic and a huge array of tiny, powered coils. The stylus contains a similar coil array, although smaller and more directionally oriented. As the stylus approaches the array, the inductance interaction between the coils creates current variations; these variations are measured and interpreted by the digitizer logic to determine pen position and point pressure (the force with which the pen’s tip is being pressed). This information is then output to a UART serial interface. The CPU receives this input on one of its DMA-mapped UART channels, and then processes the information as described later. The digitizer’s UART information is at 19200 baud, 8-bt format with no parity bits.

For debugging and programming purposes, PEN provides a terminal output via a powered RS232 serial output. The terminal runs at 115200 baud, 8-bit format with no parity bits, and is used for both output and input. PEN’s terminal interface will be described more fully in the software section; specifically, the terminal interface is intended for connection to a COM port on a PC, and has not been tested with any other device.

The CPU has DMA access to an Intel-designed USB controller; this controller interfaces the USB interface as defined in Intel’s USB standard. Because USB is not featured prominently in the current version of PEN, this document does not include significant discussion of USB operation standards.

Power is provided by the power bus illustrated in figure 2:

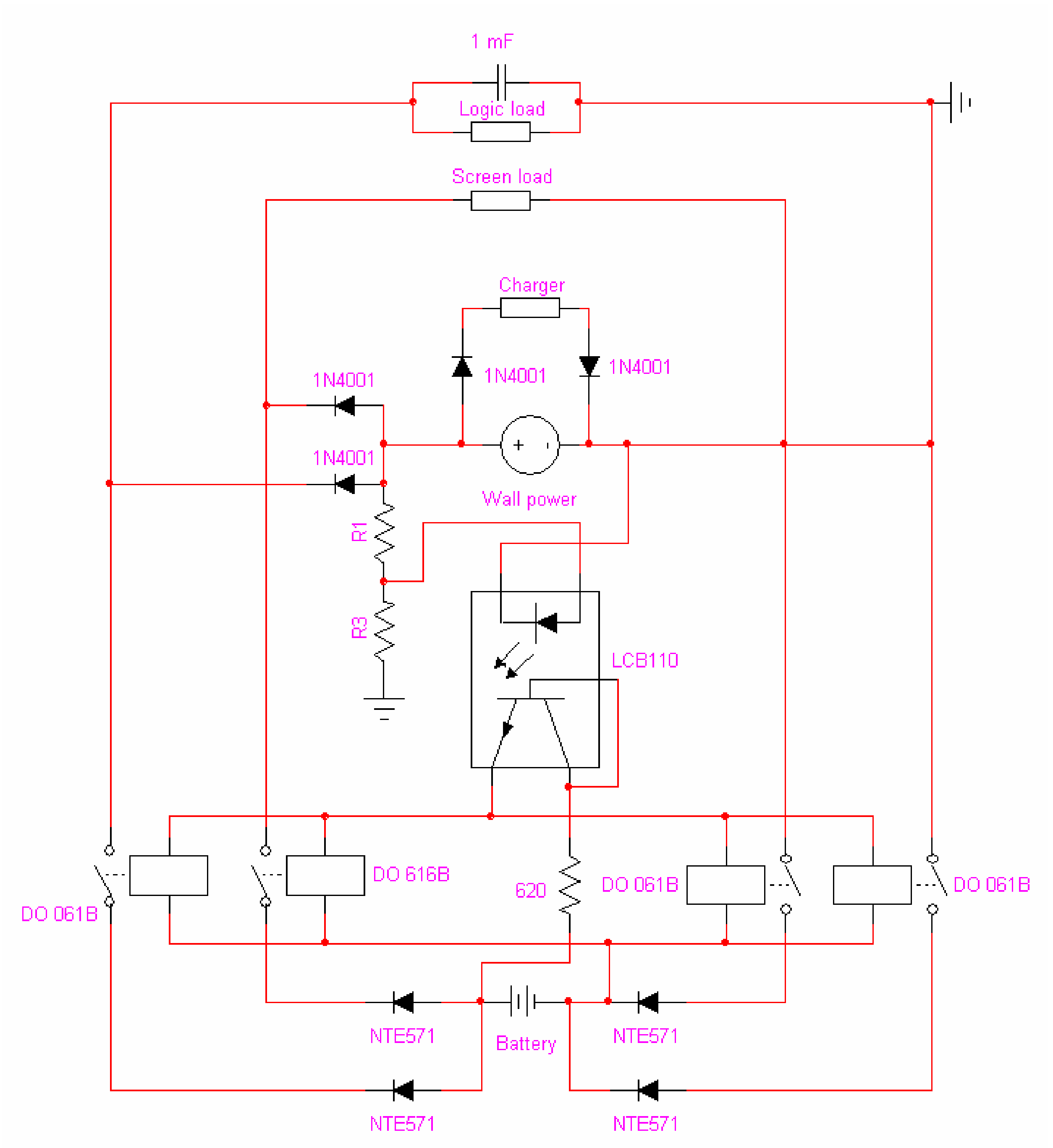

Figure 2: Power circuit schematic

PEN is designed to draw power primarily from a 3-cell 11.1VDC lithium-ion battery. This battery supplies a single 11.1V signal, which goes directly to the display (the "screen load"); the same signal also passes through a voltage regulator to produce a 5 VDC signal for use by the CPU and Gumstix components (the "logic load"). This 5 VDC signal is further reduced to 3.3 VDC by another regulator for use by the digitizer and display logic, and the LVDS converter.

To provide non-battery power, PEN has a second power input channel, intended for a 12VDC input provided by a standard AC/DC adapter plugged in to a wall socket. This "wall signal" not only powers PEN while present, but charges the battery simultaneously. To accomplish this, a specially designed lithium-ion battery charger is integrated and powered *only* by the wall signal; it has no access to the battery power circuit.

The volatile nature of lithium based batteries precludes a constant-current charging scheme, and in fact requires that the battery itself be completely isolated from the power circuit while in charge mode. To accomplish this, DC relays are used to disconnect the battery from the circuit while the wall signal is present. The battery is also constantly plugged in to the charger. Thus, when the wall signal is introduced, the battery is instantly disconnected from the circuit, and the charger activates, charging the battery; at the same time, PEN experiences only a momentary interruption in power supply due to the transfer. To counter this interruption, the "logic load" is in parallel to a high capacitance, as is visible in the schematic; this capacitance ensures that the approximately 1 ms interruption does not cause a system reset. The "screen load" is not insulated against interruption because the load is too excessive to be protected by reasonably available capacitors. The power loss does cause the screen to go blank; however, once power is restored (approx 1 ms after the blackout) the CPU sends a reset signal to the backlight, which causes it to resume normal operation.

The DC relay system pictured makes use of single-pole, single-throw relays and a single optoisolator. The optoisolator is analogous to a "normal closed" DC relay, which passes current from the battery to the DO relays. DO relays are "normal open;" in the presence of only the battery, these relays are energized and closed by the battery signal coming through the optoisolator. The battery is then connected to the main power circuit through these relays. When the wall signal is introduced, it closes the optoisolator; the loss of current from the optoisolator causes the DO relays to de-energize and open, which isolates the battery in an open circuit. The battery is then safe to charge.

PEN Software

PEN is operated by a threaded OS. This OS ensures proper display, processes digitizer input, interacts with the terminal, and manages memory. The OS runs five primary threads. PEN's thread scheduler is round-robin; no thread is weighted above any other. Hardware interrupts are preemptive over any thread, although the information produced by any interrupt may not be applicable to all threads.

Before examining PEN's threads, it is instrumental to discuss the PEN lexicon as used in this document. A *page* refers to the information corresponding to a single display, analogous to a page of paper in a notebook. Thus, when discussing the movement of pages in memory, the reader is reminded to think of blocks of 1024x768 pixel structures, and not virtual memory pages or any other page structure used in computer science. Pages are stored in three locations. The primary cache, resident in SDRAM, contains four pages in decompressed format. Decompressed pages consist of 1024x768 pixel structures, each of which contains information about that pixel's color. The LCD controller receives a pointer to one of these four cache slots, indicating where in memory it should read information for export to the screen; thus, the primary cache has one "on-screen" page and three "off-screen" pages. Pages that are not cached are stored in LZW compressed format in a FAT32 drive on the SDRAM. Finally, this FAT32 drive is mirrored onto the Flash memory for non-volatile storage.

Pages are stored in sequential order. When a new page file is created, it is assigned a random numeral for a file name; that numeral is then mapped to a page number, or sequence number. Sequence numbers are stored in the booklet.dat file, in the root of the FAT drive. When a "next page" or "previous page" event is invoked, the OS looks up the current page's sequence number, increments or decrements, and then looks up the page file associated with the resulting sequence number. That file is then displayed.

The first of PEN's five threads is the **main thread**. The main thread is responsible for the display and the processing of digitizer input. This thread executes a loop in which is reads the digitizer input buffer and translates that input into either *draw* or *keypress* events. The digitizer buffer and its processing will be discussed in detail later; draw events represent input on the "paper" area of the display, while keypress events represent input on the dashboard area. For draw events, the main thread alters the on-screen page by writing to that page's cache location. For keypress events, the thread invokes the appropriate method for the key pressed. Because the LCD controller uses a pointer to a predefined cache space location to update the screen, the main thread only needs to write pixel changes to that location to update the screen; the controller displays the changes when it automatically reads the page's contents.

The second thread is the **terminal thread**. The terminal thread scans the RS232 channel for input, parses terminal commands, and invokes the appropriate system methods. It also provides terminal echo and diagnostic output.

The third thread is the **cache manager.** The cache manager selects which pages will be decompressed and placed in the primary cache, and then decompresses those pages from the FAT drive into a cache slot. The cache manager always ensures that the two sequential neighbors of the currently viewed page are in cache, and generally uses a last-accessed replacement policy.

The fourth thread is the **FAT manager.** The FAT manager scans the cache for dirty (changed) pages. When a page is dirty, the FAT manager compresses the page into a GIF file and overwrites that page's existing GIF file in the FAT drive, thus updating the page in FAT memory to reflect its current status in cache. The FAT manager then marks the page as clean. The FAT manager handles compression in the event of a "page dump," where a dirty page must be removed from cache to make room for a desired page.

The final thread is the **Flash manager.** The Flash manager periodically compares the contents of the SDRAM FAT drive to the FAT image stored on Flash memory; when changes are found, the Flash manager erases the corresponding Flash blocks and mirrors the information from the SDRAM FAT drive onto the Flash. In this way, the Flash manager ensures that PEN's non-volatile memory is up-to-date, in the event of unexpected power loss.

PEN’s software is driven by interrupts. The Wacom digitizer’s UART channel has a dedicated interrupt to process digitizer input. The digitizer interrupt handler simply adds the stylus’ current position to a buffer of digitizer inputs, and then returns control to the thread manager. When the main thread runs, it empties as many of these buffered inputs as possible. If an input is preceded by a period of inactivity, it is considered the start of a new line; any subsequent inputs are considered to be points on that line, and the main thread connects these plotted points by drawing short, straight lines between them, forming a pen-stroke. If a line begins in the dashboard portion, a button press is detected; if that line ends (meaning the digitizer is polled and the stylus is not present) within the same button, then the main thread invokes the method corresponding to that button.

Conclusion

In designing PEN, one of the more significant obstacles faced by the team was the battery system. A Lithium-ion battery was selected for its superior energy-volume ratio and rapid charge time, but, as we worked with the battery, we discovered that the drawback to lithium’s potency is its volatility. Specifically, lithium batteries must be charged in very controlled circumstance, and are generally insulated from outside circuits by proprietary protection circuitry designed to prevent flame venting or other explosive results of improper charge or discharge. When PEN enters full production, a custom-made, professionally fabricated lithium battery will obviously be used; however, while in the design phase, a nickel-based battery chemistry would have been much more desirable.

One interesting impediment in PEN’s evolution was a side-effect due to external, long-distance wiring. The LVDS input to the LCD is a high-speed (350 MHz) serial signal; the line carries the information for each pixel’s color in rapid succession, with periodic synchronization signals provided so the screen knows which pixel is currently being transmitted. Electrically, these synch signals are represented by coordinated falling edges; the line “drops low.” The color white, represented by maximum intensity in all three LCD colors (red, green, blue), is represented (appropriately) as all 1’s. So, for a white pixel, the screen line carries a series of 1’s, an uninterrupted high signal. The spacing and length of the wires used in the prototype gave them an unwanted capacitance that would prevent the synch signal from manifesting itself: specifically, the signal could not “drop” fast enough to register as a 0, due to the stored charge in the wires. To overcome this, we redefined the color “white” as something dimmer than “true white,” e.g. a color not represented by all 1’s. This fix worked perfectly, and, with fabricated on-board wiring, the “too-white” problem will disappear.

Working with the Gumstix platform was an excellent choice. Gumstix offered an abundance of features and powerful processor at a very affordable price. However, because Gumstix is a small company, their products are less documented and rigorously tested than generally expected in the industry. A great deal of our reference material was provided by a community generated “Wiki” document, housed online. While this community was very helpful and intuitive, we often had trouble finding specific answers to simple questions, since such questions were considered so obvious as not to merit a Wiki entry.

Additionally, the Gumstix platform is a hobbyist kit by design. We worked with a single Gumstix board for 7 months with no problems or complaints. Then, with no warning, the Gumstix's voltage regulation capability disappeared, and the processor drew an excess of current and destroyed both itself and several of the interconnected components. Subsequent boards suffered from similar power-draw issues; the fourth board finally behaved as designed. Since Gumstix is a smaller company, their ability to diagnose our problem was very limited, and their readiness to replace or repair our faulty parts was even more limited.

PEN is not currently complete. Short-term features that did not make this version are:

- an external power button, and power management algorithms, triggering a "sleep" mode rather than a full power-off;
- a battery life meter, very difficult to implement in a student-lab environment with a lithium battery;
- USB connectivity, allowing PEN to act as a mass storage device for interface with a PC; and
- user accessible brightness control for the LCD.

Bill of Materiel

- 1 Gumstix basix 400xm board, from Gumstix Inc. $130
- 1 breakout gs board, from Gumstix Inc. $27.50
- 1 1024x768 SVGA LCD display and Wacom SU-015-X02 digitizer from a Gateway Tablet PC, purchase on eBay.com. $75.
- 1 EBKLVDS2 TTL-LVDS converter, from Nexxcom.com. $40
- 1 stylus compatible with Wacom digitizers, from eBay.com. $40.
- 1 11.1 V 3-cell lithium ion Winner's Circle battery, from Amondtech.com. $40.
- 1 Winner's Circle lithium ion 2-4 cell battery charger, from Amondtech.com. $60.
- 4 DO061B solid state DC relays, from Digikey.com. $9 each.
- 1 LCB110 optoisolator, from Digikey.com. $3.25.
- Miscellaneous analog components, wiring and connectors, from Digikey.com. $30

References

Most of PEN's architecture comes from the public domain. The LZW compression algorithm, ARM assembly language, FAT32, and UART and USB communications standards are all obtainable online. In the hardware design, Dr. Al Davis, Dr. Neil Cotter, Dr. Angela Rasmussen, and Chris Strong, all of the University of Utah, were especially helpful.

Working with the gumstix platform was surprisingly easy, thanks in large part to a massive online community centered at www.gumstix.org. Without this community's amassed experience, PEN would not have gotten off the ground. Similarly, in working with batteries and power supplies, www.batteryuniversity.com, a collection of battery-related articles, was instructive, and probably prevented a few lithium-ion explosions.