Evaluation and Reduction of Power Consumption in the Navigator Wearable Computer

Tom Martin

Advisor: Daniel P. Siewiorek

Master's Report

August, 1994

Abstract: The increasing density of integrated circuits has made it possible to design computers which are small and lightweight enough to be mobile. Mobile computers have constraints that their desktop predecessors do not, especially in the domains of size, weight, and power. The power consumed by a mobile computer is a key constraint, because the power determines the battery capacity required, which is a major factor in the minimum volume and weight of the system.

Power management can be performed at several levels, from the transistor level in hardware up to the application level in software. Designers of fully custom mobile computers can build in power controls at each level, resulting in a wide range of options for power management. Designers of semicustom mobile computers, using both off-the-shelf and custom components, often have fewer power management options open to them, depending upon the number of power management controls available in the off-the-shelf components.

The Navigator 1 wearable computer was a semicustom design. This paper discusses the evaluation of subsystem power consumption, describes possible modifications for reducing power consumption, and classifies the modifications along three axes. Four modifications were implemented, resulting in a factor of 2.9 increase in battery life while the system is idle. Results are also presented for a simple equation to predict battery life based upon user activity.

1. Overview

The increasing density of integrated circuits has made it possible to design computers which are small and lightweight enough to be mobile. In the commercial mobile computer market, the most widely available systems are laptop computers which closely resemble their desk-bound predecessors, with keyboards for input and large displays for output. The major difference between the two is that the laptop computers are untethered, being powered by batteries rather than connected to wall outlets. In recent years, the increasing density of integrated circuits has made possible a different form of mobile computer, the wearable computer. Wearable computers differ from laptop and desktop computers by using other means of input and output, such as speech recognition for input and miniature, head-mounted displays for output. But like laptop computers, they are battery powered, allowing them to move with the user without being tethered.

The Engineering Design Research Center has designed and manufactured three generations of wearable computers, with two new generations under development [Cogswell 92, Siewiorek 93]. Teams of electrical engineers, mechanical engineers, software engineers, and industrial designers, working concurrently, have delivered a new wearable computer approximately every 8 months since 1991. Table 1 shows the battery weight and life for the three generations of wearable computers. In each generation, the percent of system weight due to batteries is larger than that of the

	Battery Type	Battery Weight, lbs	Battery Weight, % System Weight	Battery Life, hrs.
VuMan 1	Alkaline AA	0.5	25	2
VuMan 2	Alkaline AA	0.5	50	11
Navigator	Lead acid	6.6	73	3

Table 1. Wearable computer battery type, weight, and life

previous generation. The reason for this can be seen in Table 2, which shows the major characteristics of the wearable computers. Functionality which was realized with several integrated circuits

	VuMan 1	VuMan 2	Navigator
Delivery Date	8/91	1/93	6/93
Processor/Speed	80188/8MHz	80188/13MHz	80386/25MHz
Memory	0.5MB	0.5MB, 1MB FLASH	16MB, 85MB Disk
Embedded/OS	Embedded	Embedded	Mach
Input	3 buttons	3 buttons	Speech/Mouse
Dimensions (in.)	2.5 x 5.5 x 12	1.5 x 4.0 x 4.4	3 x 8 x 10
Weight (lbs.)	2.0	1.0	9
Power (W)	3.8	1.1	16

Table 2. Characteristics of three generations of wearable computers

in VuMan 1 were combined into a single integrated circuit in VuMan 2, reducing the size and weight of the electronics. The batteries were unchanged, however, so they represented a larger fraction of the system weight.

The reason that the fraction of system weight due to batteries is higher in the third generation of wearable computer is that it has much more functionality than VuMan 1 and VuMan 2. Navigator differs from the first two generations in several major aspects. First, it uses speech as one form of input, allowing for completely hands-free operation. Second, it is a general purpose computer, running the Mach operating system. Third, it uses a modular design, allowing different configurations based upon the application needs.

The increase in functionality of Navigator over VuMan had a cost. As can be seen in Table 2, Navigator's overall weight is 4.5 times Vuman 1's, and 9 times VuMan 2's. Most of this increase is due to the batteries, as is shown in Table 1. Navigator's batteries are 13 times heavier than the batteries of VuMan 1 and 2, but Navigator's battery life is only 1.5 times VuMan 1's, and 0.3 times VuMan 2's.

A three hour battery life and nine pound weight are marginally acceptable for a wearable computer. Ideally, a wearable computer would be small enough to fit into a pocket, weigh only a few ounces, and run for months (years!) without changing batteries. A more realistic goal is to be small enough and light enough to be worn without hindering natural movement, and to run a full work day without changing batteries. Because a large portion of the weight and volume of Navigator is due to the batteries, minimizing power consumption is essential to reducing its weight and size. Reducing the power consumption allows a user to have either increased battery life for the same volume and weight, or reduced volume and weight for the same battery life.

Power consumption is also a constraint for designs other than wearable computers. Laptop computers, cellular phones, and remote data loggers are just a few examples of electronic devices which are constrained by their power consumption. Even desktop computer designers are facing tighter power constraints, after the EPA found that computers account for 5% of the U.S. commercial electrical demand and initiated the "Energy Star" program to encourage power conservation in the computer industry [Nadel 93]. Designers in these domains would also benefit from having a method for evaluating and reducing system power consumption.

This paper is a case study in the evaluation and reduction of power consumption in a computer system. Section 2 presents the initial evaluation of power consumption of Navigator. The purpose of the initial evaluation is to identify areas for reducing power consumption. Section 3 describes modifications made to reduce the power consumption. Criteria for classifying the modifications are also introduced in this section, in order evaluate tradeoffs between the power savings and other effects of a modification. Section 4 reports the results two sets of battery life measurements. The first set is the battery life with various combinations of modifications while idle 100% of the time. The second set is the battery life with various ratios of idle to active time, which is used to validate an estimate based upon the battery life will idle 100% of the time. Finally, section 5 summarizes the results. An appendix containing a detailed description of the Navigator hardware is also included.

2. Initial Power Measurements

There are several reasons why it would be desirable to know the power consumed by the various subsystems in a system. For example, knowledge of the subsystem power is important for component placement, where it is necessary to avoid "hot spots" within the design; and housing design, where the housing may be used to dissipate heat from the components. But perhaps the most important reason for portable systems is that knowing the break down of power consumption by subsystem allows one to target the subsystems with the largest power consumption when attempting to increase battery life. Very little improvement in battery life can be gained by reducing the power of a subsystem which consumes only a few percent of the system power.

In the initial stages of design, before subsystems are available, designers must depend upon specifications from the manufacturer for power consumption values. Table 3 lists the specifications and major features of the boards used in Navigator and illustrates the mismatch of requirements between desktop and mobile hardware. (Appendix A contains a detailed description of the Navigator hardware.) The design was meant to be a proof of concept, so off-the-shelf boards were used in order to minimize hardware design time and the probability of hardware design errors. Only one of the off-the-shelf boards--the CPU motherboard--was meant for mobile use. The analog-to-digital (A/D) conversion board and the Private Eye board were meant for use in desktop PC's. Neither of these boards were designed with mobility or power consumption in mind, so their power specifications were incomplete. Even the CPU motherboard's specifications could not be relied upon, as they were only for a single configuration of the board, a configuration which was different than the one used for Navigator.

Because the specifications could not be relied upon, it was necessary to measure the subsystem power consumptions. Figure 1 shows the power connections of the Navigator. (Please note that ground connections are not shown for clarity.) The system power can be found by measuring the current flowing in wires A and B, and the subsystem powers can be found by measuring the current in wires C-H. The power is simply the current flowing in a wire multiplied by its voltage.

The measurements were made with the application running, but with no inputs occurring. (For the purposes of this paper, the application running with no inputs occurring will be referred to as the

Board	Features Available	Features Utilized	Specified Power Requirements
Ampro LittleBoard 386SX	math coprocessor IDE hard drive interface up to 16MB DRAM parallel interface serial interface (2) PC 104 bus SCSI interface floppy drive controller watchdog timer keyboard connector	math coprocessor IDE hard drive interface 16MB DRAM parallel interface serial interface (1) PC 104 bus	+5V @ 1.0A
Pro AudioSpectrum 16	audio record audio playback stereo mixer CD ROM interface SoundBlaster emulation	audio record	(not specified)
Private Eye	+12V/+5Vsupply CGA mode CGA shadow mode non-CGA mode	non-CGA mode +5V supply	+12V @ 0.24A, or +5V@ (current not specified)

Table 3. Navigator off-the-shelf board characteristics

system being idle.) Two observations led to the measurements being made with the system idle. First, the average power of the unmodified Navigator was nearly the same whether it was idle or not, although the variation increased. Second, the nature of the Navigator application is such that most of the time the application is idle waiting for input. Under these conditions, the idle power consumption dominates the average power consumption.

Another reason for making the measurements with the system idle was accuracy. All measurements were made using a Fluke 45 multimeter, the accuracy of which is specified only up to 2kHz. Multimeters with a wider bandwidth exist, but were unavailable. As a consistency check, the system current was also measured by inserting a small resistance (0.5Ω) in series with the batteries (in wire J of Figure 1) and measuring the voltage across it with an oscilloscope. With the application idle, the voltage was 890mVrms, so the current in wire J was 1.78A. The current in wire J measured with the multimeter was 1.72A. The voltage when viewed on an oscilloscope

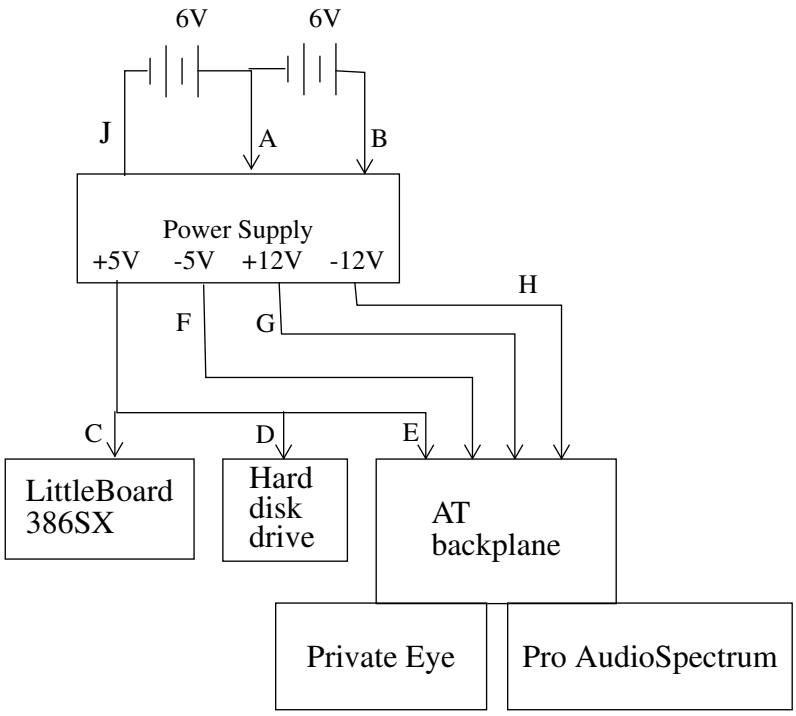

Figure 1. Navigator power schematic

was fairly constant while the system was idle, but during periods of activity there was a large variation, well above the 2kHz frequency specified for the multimeter. By making measurements with the system idle, however, problems with the accuracy were reduced. In addition, the measurements of the power were intended to be a first-order estimate of the consumption of the subsystems, in order to identify areas for savings. They were not used to predict battery life or as a measure of the benefit of a modification.

The method for evaluating power consumption was to find the system power consumption and then to find the subsystem power consumptions. Most of the difference between these two values can be accounted for by losses in the power supply, so after these measurements, the efficiencies of the power supplies were measured over a range of loads.

System and Subsystem Power Consumption

Tables 4a and 4b show the results of the system and subsystem power measurements. Measuring the system currents (wires A and B) and the +5V current for the LittleBoard and the hard drive (wires C and D) was straightforward, but it was not possible to isolate the current flowing into the Private Eye board and Pro AudioSpectrum board with both cards plugged into the AT backplane.

Because the application does not run without the Private Eye board in place, the current to the Pro AudioSpectrum board could not be measured with only the Pro AudioSpectrum board in the backplane. In order to measure them separately, the current was first measured with only the Private Eye board in the backplane, and then with both the Private Eye board and the Pro Audio Spectrum board in the backplane. The difference between the two measurements is the current to the Pro Audio Spectrum board.

Wire	Current,A	Voltage, V	Power, W
A	0.71	5.97	4.2
B	1.01	11.9	12.0
TOTAL	--	--	16.2

Table 4a. Initial system power measurements

Subsystem	+5 V Current,A	-5 V Current,A	+12 V Current,A	-12 V Current,A	Subsystem Power, W
LittleBoard 386SX	1.23	-	-	-	6.2
Hard disk drive	0.19	-	-	-	1.0
Pro AudioSpectrum	0.26	0.00	0.13	0.09	3.9
Private Eye	0.23	0.00	0.00	0.00	1.2
TOTAL	1.91	0.00	0.13	0.09	12.3

Table 4b. Initial subsystem power measurements

Subsystem	+5 V Current,A	-5 V Current,A	+12 V Current,A	-12 V Current,A	Subsystem Power, W
LittleBoard 386SX	1.23	-	-	-	7.7
Hard disk drive	0.19	-	-	-	1.2
Pro AudioSpectrum	0.26	0.00	0.13	0.09	5.7
Private Eye	0.23	0.00	0.00	0.00	1.4
TOTAL	1.91	0.00	0.13	0.09	16.0

Table 4c. Initial subsystem power consumption after including losses in the power supply.

Table 4a shows the system power consumption to be 16.2W, and Table 4b shows the sum of the subsystem powers to be 12.3W. The difference between these two can be accounted for by the efficiencies of the power supplies.

Power Supply Efficiencies

The efficiency of a power supply is defined as the ratio of its output power to its input power. The efficiency was measured for each of the supplies by placing a variable resistor between the supply output and ground. The input power was found by measuring the current in wires A and B of Figure 1, and the output power was found by measuring the current flowing through the variable resistor. Appendix A describes the power supply and the efficiency measurements in more detail.

Table 4c shows the power consumed by the subsystems after taking into account the losses in the power supply. For example, the LittleBoard uses 1.23 x 5 = 6.15W from the +5V supply. Dividing this by the +5V supply efficiency of 0.8 gives the LittleBoard's consumption after accounting for losses in the power supply, 6.15W / 0.8 = 7.7W. The total for the subsystem power is now within 0.2W of the total system power. This difference can be accounted for by losses in the wires in the system and measurement errors.

The evaluation of the power consumption of the subsystems shows that the two largest consumers are the LittleBoard and the Pro AudioSpectrum board, which account for 47% and 35% of the system power, respectively. Any modification which does not significantly reduce these two consumers will result in only a minor increase in battery life.

3. Modifications: Classification and Description

Power management can be performed at several levels, from the transistor level in hardware up to the application level in software. In a custom design, designers can build in controls at each level, resulting in a wide range of options for saving power. Because Navigator was not a custom design, there were few options available for saving power. Two of the three boards, intended for use in desktop PCs, had no power management modes. The operating system, designed for a workstation, did not take power consumption into account when managing resources nor did it offer calls to utilize the power management options that were available. The application, with no

controls available from the levels below, did not even attempt to manage power. However, even without a large number of options, it was possible to substantially increase the Navigator battery life, as this section and the next will show.

3.1 Classifications

There were a few modifications that could be made to save power, in both hardware and software. Each of these modifications can be classified along three axes:

1. Transparency to user--whether or not the modification interferes with a user's interaction with the application.

2. Dependence on user activity--whether or not modification saves power regardless of the type or frequency of user actions.

3. Amount of increase in battery life--measured with respect to the unmodified, idle system.

A modification which is transparent to the user, independent of user activity, and greatly increases battery life should obviously be incorporated into a design. But for a designer faced with a decision about whether or not to incorporate one which is visible to the user, dependent on user activity, and only slightly increases battery life, the choice is much less obvious. Is the increase in battery life enough to offset the lack of transparency? Will the modification increase battery life at all with normal user activity? These are the power tradeoffs that must be made, in addition to the normal tradeoffs of cost and reliability.

Table 5 lists six possible Navigator modifications identified during the initial evaluation of power consumption, ranging from purely hardware modifications at the top of the table, to modifications under operating system control, to modifications under user level code control at the bottom. Each of the modifications is classified according to the above axes. The battery life column will be explained in Section 4.

Two trends are apparent in Table 5. The first trend is that hardware modifications are transparent, while those requiring software control are not. Two counterexamples show that this trend is not strictly true: One could imagine a hardware-only change that dims or turns off the backlight for a

Area	Transparent?	Activity Dependent?	Increase in Battery Life, min
Power Supply: +12V	Yes	No	100
A/D Board: SCSI termination	Yes	No	20
CPU: Slow clock	No	Yes	--
Halt during idle loop	Yes	Yes	130
Hard Drive: Standby	No	Yes	--
Standby with timer initialized	No	Yes	40

Table 5. Characteristics of six possible modifications

display when the battery voltage drops below some value, which would certainly not be transparent to the user. One could also imagine a piece of user code in a tight loop waiting for input which turns off some unused hardware until the input occurred, which would be transparent to the user, assuming that the hardware could be powered up unnoticeably.

The second trend in Table 5 is that hardware modifications are not activity dependent, while those requiring software control are. Counterexamples to the hardware portion of the trend include integrated circuits on the market today which, without software control, turn off unused buses. The ability of these circuits to save power depends on which devices were accessed and how often, making them activity dependent. The software portion of this trend, on the other hand, seems to be true. Software can reduce power by putting devices into low power modes, or perhaps even by using sequences of instructions which have been shown to use less power than other sequences of instructions. In the first case, power can only be saved while the devices are not in use, and in the second case, power can only be saved while the sequence is being executed. Both of these cases are activity dependent.

The rest of this section describes the implementation details of the modifications, beginning with the hardware, then moving to the Mach microkernel code, and finally to the user level code.

3.2 Modification Descriptions

3.2.1 Hardware Modifications

Power Supply

The power supply schematic is shown in Figure 2. The +12V supply was found to be 75% effi-

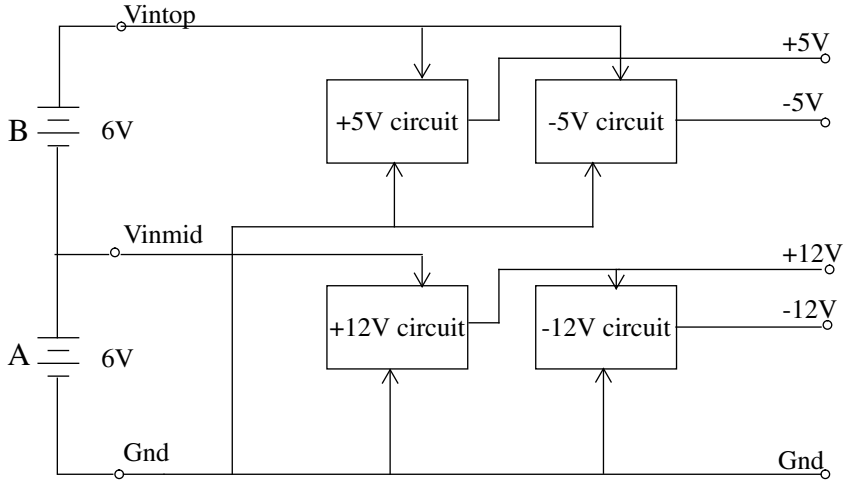

Figure 2. Unmodified power supply schematic

cient in the original design, with an input voltage of approximately 6V. The efficiency was found to increase with increasing input voltage, leveling off at nearly 90% for input voltages of 9.5V to 11.5V, as can be seen in Figure 3. Because Vintop is greater than 12V for most of the battery life, it could not be used as the input to the +12V supply, which has a maximum input voltage of 11.5V. Placing two diodes between Vintop and the +12V supply input, as shown in Figure 4, lowered the input voltage to the allowable range. Using Vintop minus two diode drops instead of Vinmid increased battery life due to two effects. First, the higher input voltage increased the efficiency of the +12V supply. Most of the power reduction due to the increased efficiency was offset by the power dissipated by the diodes. Second, equally loading the batteries results in a larger total amount of energy available to the system. With the unmodified circuit, battery A discharges more quickly then battery B, so at the end of battery A's discharge cycle, battery B still has energy left. Continuing to discharge the batteries beyond this point will damage battery A due to deep discharge.

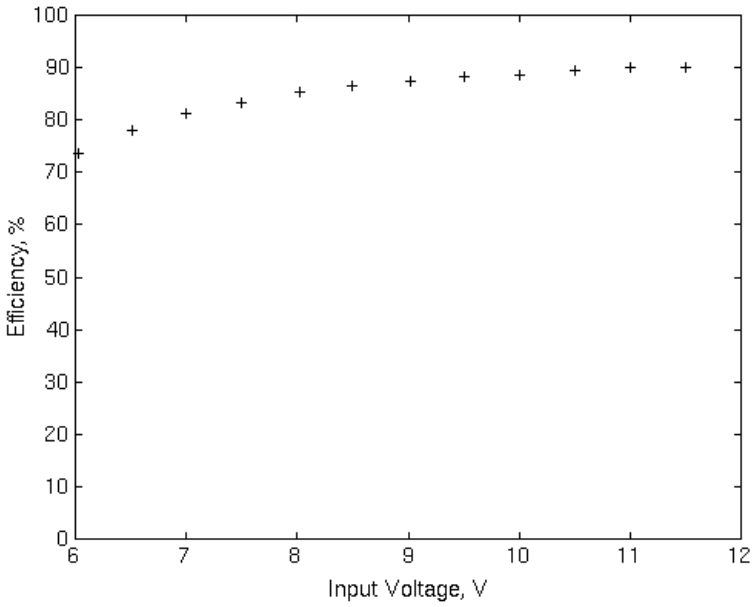

Figure 3. +12V Efficiency vs. Input Voltage

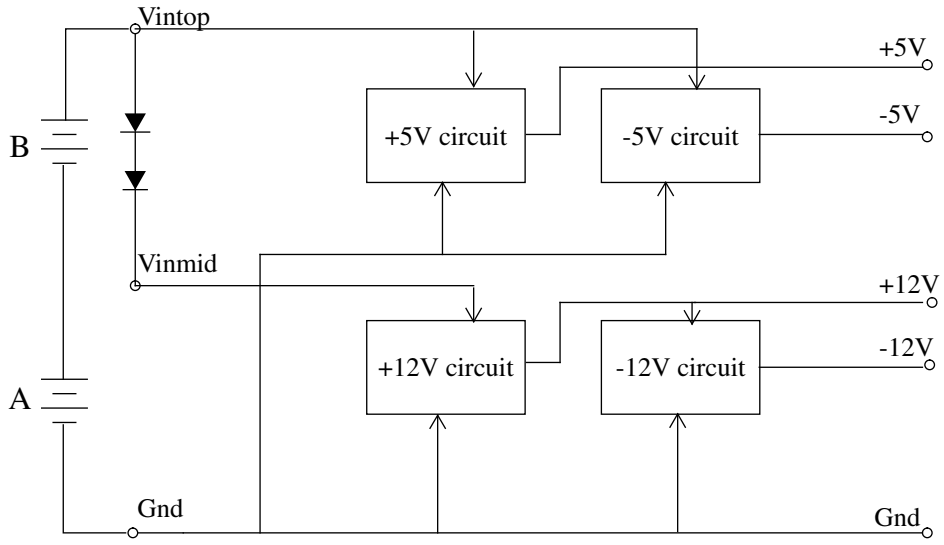

Figure 4. Modified power supply schematic

Pro AudioSpectrum

The Pro AudioSpectrum board has a SCSI interface for use with CD-ROM drives. The SCSI signals are terminated with resistors, resulting in the static dissipation of 0.8W. Because the terminating resistors are only needed when the interface is used, and because the Navigator did not need

the interface, the resistors were removed. Accounting for the efficiency of power supply, the removal resulted in an overall savings of 1.0W.[1]

3.2.2 Mach Microkernel Modifications

The CPU

The system is usually idle while the Navigator application is running. Except for a few background processes which require a small number of cycles, the CPU is active only for a short period after user input, be it speech or mouse, and then is idle until the next user input. Measurements of the dynamic power consumption of the unmodified Navigator as it executed code showed that it was the same or lower while the system was active as compared to when idle. The power consumed executing the idle loop is wasted.

The first attempt at saving power while idle was to take advantage of a feature of the LittleBoard that changes the CPU speed from 25MHz to 8MHz, resulting in a savings of about 3W. The intent was to lower the CPU speed until the user generated some input, raise the CPU speed while responding to the input, and then lower the speed again once the response was complete. The problem with slowing the clock is that interrupts are handled at the slow clock speed. The cursor movements, for example, became unacceptably jumpy with the CPU slowed down. The fix is to modify the interrupt handler to check the CPU speed before servicing the interrupt, raise the speed if necessary, and then lower the CPU speed when the service routine completes.

A member of the Mach group suggested a much cleaner solution: Inserting a halt instruction into the idle loop. The “hlt” instruction in the x86 architecture halts the CPU until an interrupt is raised. One of the advantages of this solution is that the change is transparent to the user. Interrupts are serviced at full speed, and when the processor is not idle, code execution occurs the same as without the change.

The modified idle loop code executes as follows: As long as there are no threads waiting to run and the run queue counts are zero, the processor checks for asynchronous system traps. Once the

1. The termination resistors for the LittleBoard 386SX’s unused SCSI interface were also removed, with a similar savings, but the removal occurred a year before these measurements. All measurements in this paper were made with the LittleBoard’s termination resistors removed.

"hlt" instruction is executed, the CPU will remain halted until the next interrupt occurs. Under Mach 3, the clock interrupt happens every 10ms, so the CPU will be halted at most 10ms before checking to see if the processor state has changed. By halting the CPU in the idle loop, power consumption is reduced by 4W when the Navigator is idle. Because of the large difference between idle and active power, the battery life with the modification depends on the ratio of idle to active times. Section 4 gives the results of experiments with different ratios of idle to active times.

Hard disk drive

The drive has three power management modes: active, standby, and sleep. In standby, the disk is spun down, and the drive needs only to be accessed to return to active mode. In sleep, the disk is spun down, but the drive needs to be reset in order to return to active mode. The drive can go to standby due to either a power management command or a signal from an on-disk timer. The timer decrements every 5 seconds when enabled and resets to its initial value every time the drive is accessed. The power management commands and their effects are given in Table 6 [Integral 92].

The power management commands which initialize the timer (E2, E3) greatly simplify the software for putting the disk in standby mode. Without them, the software would have to monitor disk activity and send an E0 command whenever the disk should be shut down. With these two commands, the software need only to initialize the timer to an appropriate time-out period by writing the sector count register, and then send one of the two commands. The drive then goes to standby whenever it has not been accessed for the time-out period, without further software intervention.

The Mach kernel was modified to take advantage of the standby mode using commands E0 and E2. Three routines in the hard disk drive driver were changed: hdsetstat(), hdstart(), and hdintr(). Hdsetstat() handles I/O control operations, such as setting drive parameters, hdstart() initiates the drive I/O request at the front of the current queue, and hdintr() services the drive interrupts. Chapter 8 of [Leffler 89] gives a more detailed description of the interaction of these routines. Hdsetstat() was modified so that when it receives power management requests, it allocates a block I/O buffer, sets the buffer's operation field to the power management command, and then calls hdstrategy(), which queues disk I/O requests so that they are serviced in an efficient manner.

Command	Effect
E0	The drive will go immediately to Standby.
E1	The drive will go immediately to Active.
E2	The drive will go immediately to Standby. If the sector count register is zero, then the timer will be disabled. If the sector count register is non-zero, the timer will be enabled and initialized with the sector count value.
E3	The drive will go immediately to Active. If the sector count register is zero, then the timer will be disabled. If the sector count register is non-zero, the timer will be enabled and initialized with the sector count value.
E5	If the drive is in Active mode, the sector count register will be set to hex FF. If the drive is in, going to, or recovering from Standby mode, the sector count register will be set to 00.
E6	The drive enters Sleep mode. Either a software or hardware reset is required to recover from this mode. Upon receiving a reset, the drive will enter Standby.

Table 6. Hard disk drive power management commands.

Hdstart() was modified so that it recognizes the power management commands in the buffer and initiates them correctly. Hdintr() was modified so that it handles interrupts caused by the drive receiving power management commands.

The other power management commands could be used by simply adding case statements for each one in both hdsetstat() and hdstart(). The only reasons they were not added to the code is that they were not necessary for Navigator.

The power consumed by the drive in active mode is rated at 1.0W; in standby, 0.035W; and in sleep, 0.010W. The time to recover from standby is typically 1.5s and the drive consumes an average of 2.0W during that time. The drive is rated at 1,000,000 starts/stops.

Using the typical values from above, the drive needs to be in standby at least 2.3 seconds in order

for there to be a power savings. Using worst case values (minimum idle power, maximum standby power and standby recovery time), the drive needs to be spun down 5.3 seconds for there to be a power savings.

3.2.3 User Level Code Modifications

The application code required only one minor change. It was modified to call a disk standby program as soon as the application has finished loading its main screen. The disk standby program initialized the disk timer to 1, and then sent the E2 command described in the previous section. The drive will then go to standby whenever 5 seconds passes without a disk access.

In order to make this more effective, the program "update", which performs a sync on the file system periodically, was not run. Usually update is started up just after the operating system boots. With update running, the drive wakes up every 30 seconds even if there are no other disk accesses. Because of the cost of spinning up the disk, spinning up every 30 seconds would half the power savings due to putting the disk in standby.

4. Results

Two sets of experiments were performed. The first set was a series of battery life measurements with different combinations of the modifications described in Section 3. The second set was a series of battery life measurements with various ratios of idle to active times.

4.1 Battery Life vs. Modification Measurements

The purpose of the first set of experiments was to determine the actual effect of the modifications on battery life. Current measurements like those of Section 2 indicated the likely effect of the four modifications, but they were somewhat suspect due to the bandwidth of the multimeter. Because extending Navigator's battery life was the main purpose of this project, the battery life was used to more accurately gauge the effect of each modification. The measurements were made with the system idle in order to obtain an upper bound on the battery life. These idle battery life values can be used to predict the battery life for different ratios of idle to active time, as will be shown in Section 4.2.

Finding the battery life for every combination of the four modifications would have been too time consuming. With three replications for each combination in order to estimate the experimental errors, and given the recharge time of the batteries, the experiments would have taken 48 days. Instead, a fractional factorial experimental design was used, with three replications of eight combinations. The combinations were not chosen randomly, but were chosen such that the effect of each modification could be calculated from the results. As can be seen in the first column of Table 7, each modification is present in half the experiments, and every pair of modifications is present in a quarter of the experiments.

Modifications	**Average Battery Life, minutes**	**Measured Power, Watts**
None	158	16.2
Idle loop, +12V supply	359	11.3
Hard drive in standby, +12V supply	262	14.5
Idle loop, hard drive in standby	302	10.3
SCSI termination, +12V supply	249	14.3
Idle loop, SCSI termination	262	10.6
Hard drive in standby, SCSI termination	185	13.9
Idle loop, hard drive in standby, SCSI removed, +12V supply	451	9.1

Table 7. Battery life and power for several combinations of modifications

With this experimental design, the battery life of the system is modeled as a linear combination of variables representing whether or not the modification is present and variables which represent the change in battery life due to the modification:

$$y = q_0 + q_a x_a + q_b x_b + q_c x_c + q_d x_d + q_{ab} x_a x_b + q_{ac} x_a x_c + q_{bc} x_b x_c + e$$

where q_0 is the average battery life of all the experiments, q_i represents the change in battery life due to modification i, and x_i is -1 if modification i is not present and 1 if it is. The q_{ij} terms represent the change in battery life due to the interaction of modifications i and j. The e term is the

experimental error. Experiments with eight combinations of modifications allowed the eight q terms in the above equation to be determined. Repeating each experiment three times allowed the experimental error term, e, to be estimated.

Table 8 lists the percentage of variation explained by each modification, as well as that explained by the interactions between modifications and experimental error. The variation, also referred to

Modification	**Variation explained, %**
+12V	34.5
SCSI termination	0.9
Idle loop	54.4
Hard drive in standby	6.0
Interactions	3.2
Experimental errors	1.1

Table 8. Variation explained by each modification

the Total Sum of Squares (SST), is a measure of how much each experiment differed from the average of the experiments, and is equal to $\sum(y_i - \bar{y})^2$. Modifications which explain larger percentages of the variation are more important, while those which explain smaller percentages are less important. (Jain [Jain 91] contains a detailed description of fractional factorial experiments, and the method for calculating the effect of a variable in such an experiment.) The idle loop and +12V modifications account for nearly 90% of the variation observed during the eight experiments, and thus had the greatest effect on battery life. The other modifications, the interactions between modifications, and the experimental errors account for roughly 10% of the variation, and can be considered negligible.

An interesting item to note is that row two of Table 7 has a higher power AND a higher battery life than row four. The reason is the +12V supply change. Without the change, the battery life is determined by battery A in Figure 2, because the current drawn from A, I_a, is higher than the current drawn from battery B, I_b. Assuming the batteries have energy E and voltage V, then the battery life $L = E/(VI_a)$. The energy removed from battery B is $(VI_b)*L = (VI_b) * E/(VI_a)$. Cancelling

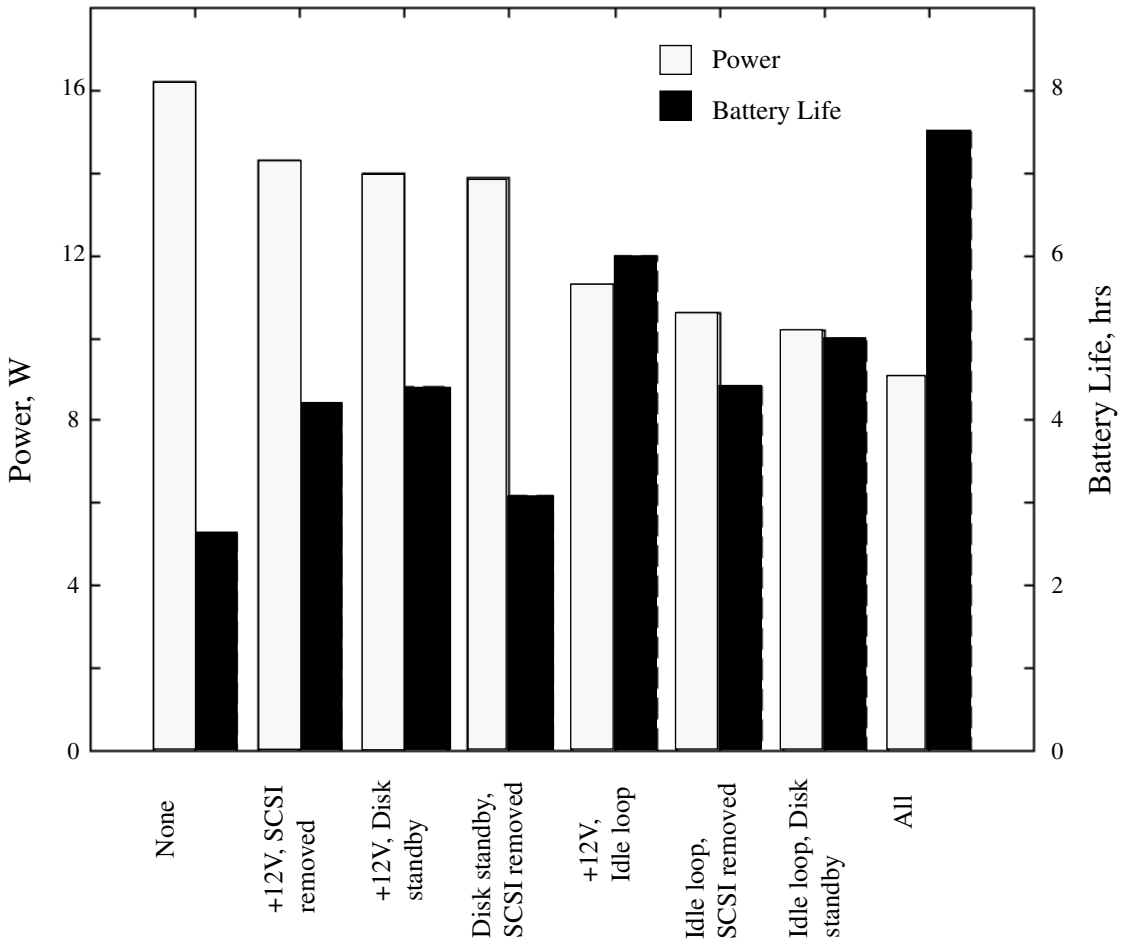

Figure 5. Power, Battery Life for Several Configurations

the *V*'s and rearranging, the energy removed from battery B is $E * (I_b/I_a) < E$. So the total energy removed from the two batteries during the battery life is less than 2*E*. With the +12V supply change, the current drawn from battery A is equal to the current drawn from battery B. Now the total energy removed from the two batteries is 2*E*. So the reason the battery life is longer with a higher power is that there is more energy available. This is an example where having a benchmark that measured only the power coming into the system would give incorrect results if the goal were to increase the battery life.

Figure 5 shows the battery life and the power for the different configurations, sorted in order of decreasing power. In each case where a configuration has a higher power and a longer battery life than another configuration, the higher power configuration has the +12V change and the lower

power configuration does not.

The rule of thumb followed during design was “Changes which lower power are better”. But that rule should be amended to “Changes which lower power are better, so long as they do not decrease the amount of energy available”.

The results in Table 7 show that modifying the +12V supply and the idle loop increases battery life by more than 200 minutes over that of the unmodified Navigator (row two), while removing the SCSI termination and placing the hard drive in standby increases the battery life by less than 30 minutes over the unmodified Navigator (row seven). All the modifications together increase the battery life by 300 minutes, a factor of 2.9 increase over the unmodified Navigator. This increase is obtained only when the Navigator is idle 100% of the time, as will be shown in the next section.

4.2 Battery Life vs. Idle/Active Ratio Measurements

Section 3 explained how some modifications save power all the time, while others are activity dependent. For Navigator, the activity dependent modifications can be viewed as saving power only while the system is idle, so the battery life will be determined by what fraction of time the system is in use. Assuming the energy capacity of the battery is constant for different loads, the battery life as a function of the fraction of time the system is idle is given by

$$BatteryLife = \frac{Energy}{(1 - F_{idle}) \times P_{active} + F_{idle} \times P_{idle}}$$

where P_{active} is the power while active, and P_{idle} is the power while idle. Letting $P_{active} = Energy/BatteryLife_{active}$ and $P_{idle} = Energy/BatteryLife_{idle}$, and rearranging,

$$BatteryLife = \frac{BatteryLife_{active}}{1 - F_{idle} + \frac{F_{idle}}{\left(\frac{BatteryLife_{idle}}{BatteryLife_{active}}\right)}}$$

The above equation is the same as Amdahl’s Law [Hennessy 90]. It shows that the increase in battery life due to a power saving modification is limited by the fraction of the time the modification

can be used.

The equation was verified by writing a script which was a mix of application operations and "sleep" statements, timing the script to find the ratio of idle to active time, and then measuring the battery life while running the script continuously on a Navigator with all four modifications. Assuming that whenever the machine is active the CPU is not halted and the drive is not spun down, then the $BatteryLife_{active}$ would be that of the "SCSI termination, +12V supply" row of Table 7, 249 minutes. The $BatteryLife_{idle}$ would be 451 minutes. Figure 6 shows the measured battery life for several ratios of idle to active time versus the predicted battery life.

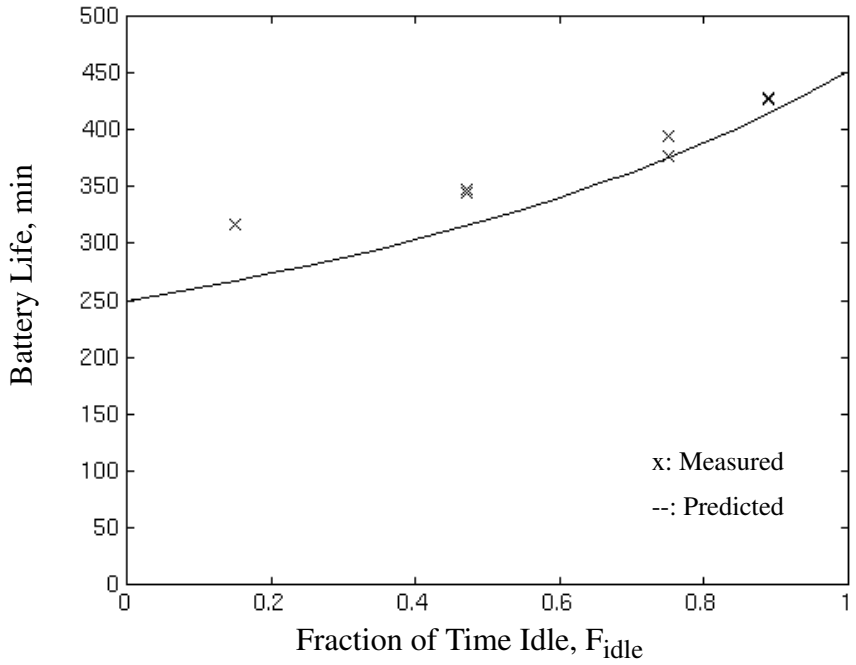

Figure 6. Predicted, Measured Battery Life vs. Fraction of Time Idle

The predicted values are consistently less than the measured values. This can probably be explained by the assumption that when the Navigator is active, the battery life is that of the "+12V, SCSI termination" row of Table 7, meaning that the disk is spun up. The disk is not necessarily spun up whenever an operation is performed, so using the "+12V, SCSI termination" is pessimistic. A more accurate calculation would include both the case when the system is active with the disk spun up and the case when the system is active with the disk in standby.

The results look nearly linear, so one might conclude that a reasonable approximation would be a line from $BatteryLife = BatteryLife_{active}$ at $F_{idle} = 0$ to $BatteryLife = BatteryLife_{idle}$ at $F_{idle} = 1$, or

$$BatteryLife = (BatteryLife_{idle} - BatteryLife_{active})F_{idle} + BatteryLife_{active}$$

$$= (BatteryLife_{idle})F_{idle} + BatteryLife_{active}(1 - F_{idle})$$

Letting $BatteryLife_{active} = Energy/Power_{active}$ and $BatteryLife_{idle} = Energy/Power_{idle}$, and dividing both sides by *Energy*, then

$$1/Power = F_{idle}/Power_{idle} + (1-F_{idle})/Power_{active}$$

This does not make sense physically, whereas the earlier equation was derived from basic physical principles. The underlying physics, then, argues against a linear estimation for the battery life as a function of fraction of time idle.

The operations in the script were probably representative of how someone might use the application. All of the commands of the application were used, and each of the databases was accessed. The relative frequency of the operations, however, was chosen arbitrarily rather than on the basis of a quantitative user study. What is important to note is that, given the results of a user study for an application, the above equation for battery life, and the battery life for different modes of operation, a designer could estimate the battery life for the application.

5. Summary

The power consumption of the Navigator wearable computer has been evaluated in order to increase the battery life. The power consumption of each of the subsystems was measured. The method for measuring can be employed with systems other than Navigator, such as a desktop PC. Possible modifications to each subsystem were considered, and classified along the axes of user transparency, activity dependence, and increase in battery life, in order to have a better understanding of the tradeoffs involved.

Finally, two series of battery life experiments were conducted. The first set of experiments resulted in estimates of the increase in battery life to be expected from each modification. These experiments showed that two of the modifications--changing the +12V supply circuit and halting the CPU during the operating system's idle loop--resulted in a factor of 2.3 increase over the

unmodified battery life while the system was idle. Two additional modifications, removal of the SCSI termination and placing the hard disk drive in standby, resulted in a factor of 2.9 increase over the unmodified battery life while the system was idle.

The second set of experiments confirmed a simple equation which allows a designer to estimate the battery life of an active system knowing the fraction of time the system is idle, the battery life while 100% idle, and the battery life while 100% active. The equation shows that the amount of increase of battery life due to some modification is limited by the fraction of time the modification can save power.

Power consumption will become more of a constraint in the design of computers, as the popularity of such devices as portable PCs and personal digital assistants grows, and as designers of desktop computers attempt to make their systems more power efficient in order to reduce the operating costs. This case study has shown that a systematic evaluation of power consumption can reveal areas for great reductions. As designers become more aware of these areas for simple savings, a systematic evaluation of power consumption will still be important, as it will be a necessary tool for identifying more complex techniques for power management, such as changes in the operating system.

Bibliography

[Cogswell 92] B. Cogswell, Z. Segall, D. Siewiorek, and A. Smailagic, "Wearable Computers: Concepts and Examples", CDS Research Report, Carnegie Mellon University, December, 1992.

[Hennessy 90] J.L. Hennessy and D.A. Patterson, "Computer Architecture: A Quantitative Approach", Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[Integral 92] Integral Peripherals, "Product Manual for the McKinley 1885", Integral Peripherals, 1992.

[Jain 90] R. Jain, "The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling", John Wiley and Sons, Inc., New York, NY, 1991.

[Leffler 89] S. J. Leffler, M. K. McKusick, M. J. Karels, and J. S. Quarterman, "The Design and Implementation of the 4.3BSD Unix Operating System", Addison-Wesley Publishing Company, Inc., Reading, MA, 1989.

[Nadel 93] B. Nadel, "The Green Machine" PC Magazine, May 25, 1993. p110-145.

[PowerSonic 93] PowerSonic Rechargeable Battery Catalog, PowerSonic Corp., 1993.

[Siewiorek 93] Siewiorek, D., Smailagic, A., Lee, J., and Tabatabai, A. "An Interdisciplinary Concurrent Design Methodology as Applied to the Navigator Wearable Computer System", EDRC Technical Report, Carnegie Mellon University May 1993.

Acknowledgments

This material is based upon work supported under a National Science Foundation Graduate Research Fellowship, the Engineering Design Research Center, and the Advanced Research Projects Agency. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the National Science Foundation, the Engineering Design Research Center, or the Advanced Research Projects Agency.

I would like to thank Bob Baron, Barry Eisel, Clauss Strauch, David VanRyzin, and Jim Zelenka

for their technical assistance, and Jim Beck, Forrest Chamberlain, Janeen Deang, Grace Ho, and Jason Lee for answering every question they could.

Appendix A: Description of Navigator Hardware

The CPU motherboard is the LittleBoard 386SX from Ampro Corporation. The board is essentially a PC AT, containing an AMD 25 MHz 80386SXL, an 80387 math coprocessor, 16 MB of DRAM, a SCSI drive interface, an IDE drive interface, a floppy drive interface, two serial ports, a parallel port, and a keyboard interface. Adding an AT backplane allows up to two additional boards to be connected. In the usual configuration, the backplane holds an A/D board and a Private Eye controller board. The Navigator also has an 85MB IDE hard disk drive, the McKinley 1885 from Integral Peripherals.

The A/D board is the Pro AudioSpectrum 16 from Media Vision. The board has many features, but Navigator used it only for 16 bit, 16 KHz, analog-to-digital conversion. Unused features include SoundBlaster compatible output, stereo audio mixing, and SCSI CD-ROM drive support.

The Private Eye board is supplied by Reflection Technology with their head mounted display. The board allows the Private Eye to be used with a PC. Jumpers control the range of addresses used by the board, the mode (CGA or non-CGA) of the screen, and source of the display power.

One of the custom boards provides the power supply and speech detection circuitry. The power supply generates +/- 5V and +/- 12V from two 6V batteries. The speech detection circuit is necessary because the speech recognition software runs at eight times real time on Navigator, so the unit cannot continuously monitor the microphone for the onset of speech. The circuit generates an interrupt when a user begins speaking and another when the user stops. The first interrupt signals that the unit should begin A/D conversion of the microphone input, and the second signals that the unit should stop A/D conversion and begin recognizing the speech. The interrupt routines have not yet been integrated with the Navigator application software, which currently has the user click on a button just before speaking and again just after speaking.

The other custom board is the AT backplane. The board's only components are a connector for the LittleBoard's PC 104 bus and two edge connectors for PC AT boards.

Power Supply Efficiencies

The Navigator power supply board generates +5V (4.0A), +12V (0.25A), -5V (0.10A), and -12V

(0.10A). All current ratings are maximum. Both the +5V and +12V supplies use integrated switching regulators from Power Trends, Inc. These regulators need only an external diode and two capacitors, resulting in a small board area for the supplies. The +5V regulator is a step-down regulator, with an input voltage range of 7.3V to 20.0V. The +12V regulator is a step-up regulator, with an input voltage range of 4.5V to 11.0V. Each has a typical efficiency of about 80% over a wide range of loads, as depicted in Figures A.1 and A.2.

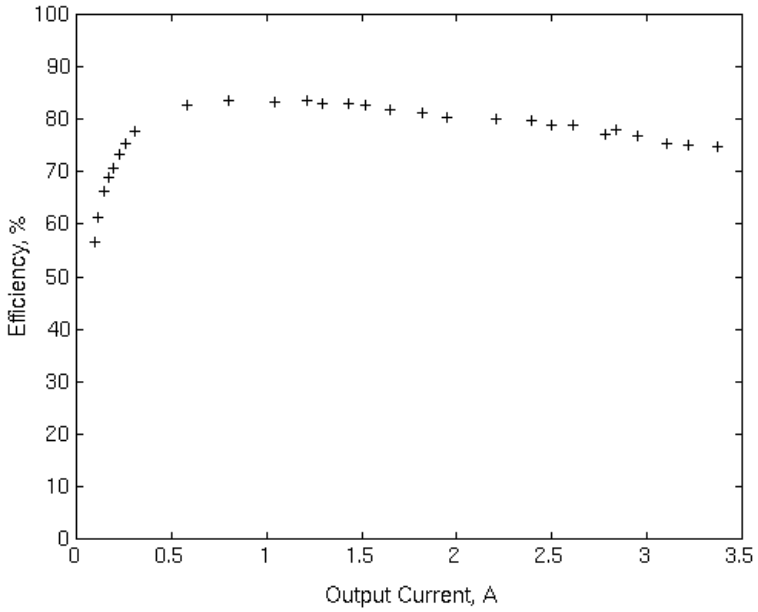

Figure A.1. +5V Efficiency vs. Output Current (maximum 4A)

The -5V and -12V supplies are more complicated circuits. The -12V supply uses the LT1054 switching converter in a simple inverting configuration without regulation. The output at the expected load of 80mA is nominally -11.1V, which is within the allowable -12V +/- 10% range. The efficiency of the -12V supply over a range of loads is shown in Figure A.3. Note that, because the input power was measured at the inputs to the entire power supply board (wires A and B of Figure 1) and because the -12V uses the output of the +12V supply as its input, the efficiency shown is actually the efficiency of the +12V circuit and the -12V circuit combined, i.e. if the +12V circuit and the -12V circuit efficiency were both 80%, then -12V efficiency shown would be 80% x 80% = 64%.

The -5V efficiency is shown in Figure A.4. The -5V supply uses the LT1054 in an inverting con-

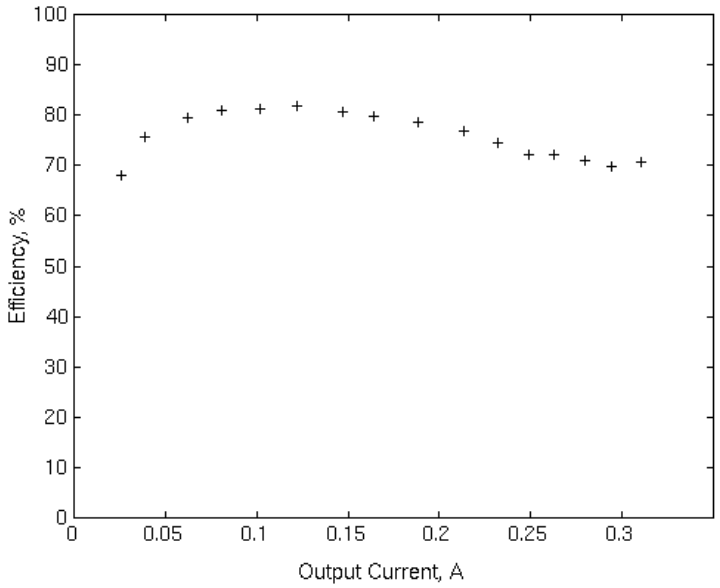

Figure A.2. +12V Efficiency vs. Output Current (maximum 0.25A)

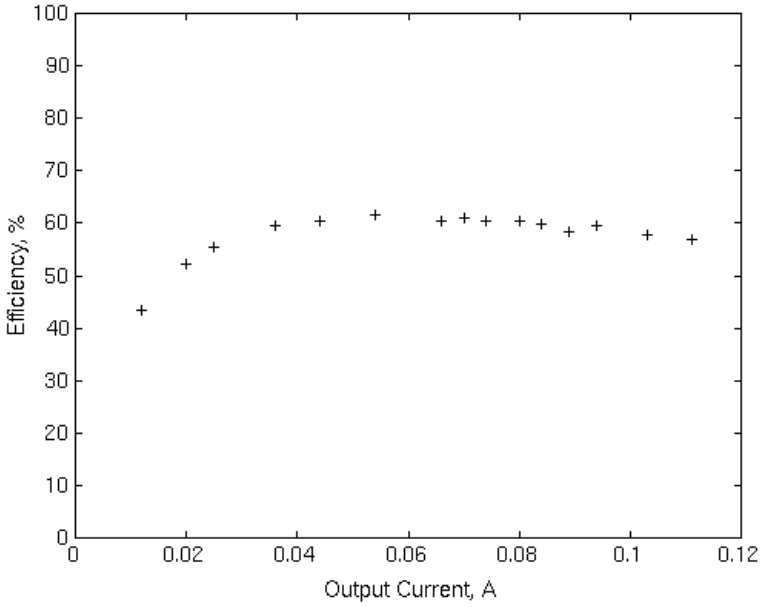

Figure A.3. -12V Efficiency vs. Output Current (maximum 0.1A)

figuration with regulation, which results in a low efficiency (about 30%). This configuration is necessary to generate -5V +/- 5%.

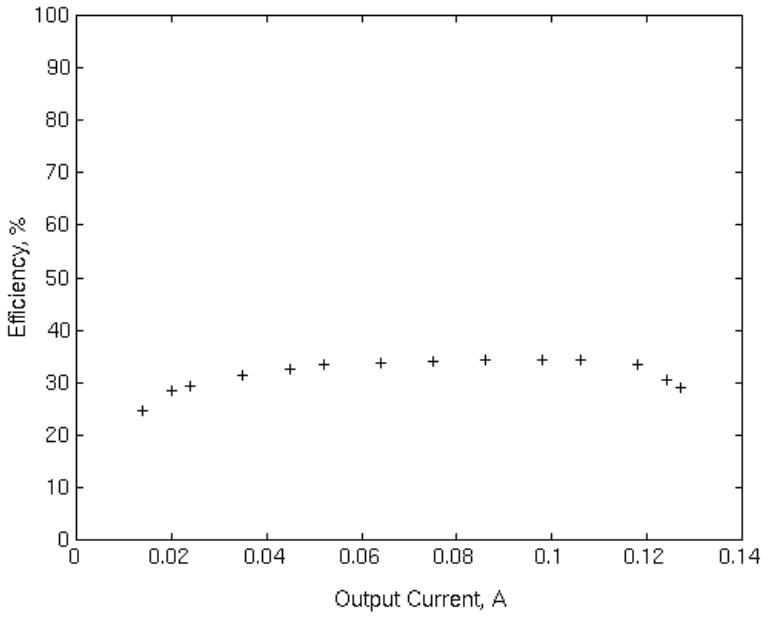

Figure A.4. -5V Efficiency vs. Output Current (maximum 0.1A)