Voltage boost and buck circuits using Atmel AVR Tiny13V for driving a white LED.

By Steven Weber, KD1JV 1/26/09

The PWM feature of the Tiny13 processor can be used to make a simple voltage boost or buck circuit to power a White LED from either 1.5 volt battery (boost) or a 6 volt supply (buck). Voltage regulation is not included, as it is not important in this application, though it could be added using the on chip voltage comparator or A/D to control the PWM duty cycle. But because there is no regulation, it is not a good idea to run this circuit without a load. The use of the processor instead of a switching regulator chip allows the easy addition of push buttons for on/off , brightness control and possibly strobing. Plus it is instructive to do this task in software which is likely better done in hardware.

Voltage boost for 1.5V battery:

One disadvantage to using the processor to boost a 1.5 V supply is that it requires it's own 3V supply to operate. The main reason for this is to get a drive voltage to the MOSFET switch high enough to ensure a low on resistance. The switching FET used here has a typical gate thrush hold voltage of 1V, but requires at least 2 volts to turn it on to a reasonably low resistance state. The second reason for using a 3V supply is the Tiny13V is specified to work down to 1.8 V, which is just a little higher voltage than a fresh alkaline battery supplies. Since the processor draws so little current, good battery life can be obtained from a small lithium coin cell. To ensure the processor draws as little current as possible, it is set to a power down mode when the LED is off and an idle mode when the LED is on. During idle mode, only the PWM counter and clock is running. In power down mode, all clocks are turned off and the battery will not even notice it is there.

While voltage regulation could be added, the LED regulates the voltage to some extent and varies with current due only to its ESR. The variation in light output with supply voltage isn't enough to warrant the added complexity of voltage regulation and would only hasten the demise of the battery as it increased the current as the voltage goes down. Using just the fixed PWM duty cycle, the LED will light with as little as 100 mV to the LED supply.

Boost regulators work by “charging” an inductor with a current. When the current is turned off, the magnetic field collapses causing a voltage to develop across it which is greater than what was used to charge the inductor. A diode routes this voltage into the load and a filter cap. The switching frequency, on/off time ratio and desired output voltage and current determines the value of the inductor needed.

Calculating the inductor value needed for the boost circuit:

The most important part of the circuit is knowing the proper inductor value to use. A good white paper on designing switching regulators can be found at ON semi (http://onsemi.com) application note AN920/D. Basically, knowing the input voltage, desired output voltage and current, saturation voltage of the switching FET, forward voltage drop of the rectifier diode and the switching cycle time, we can calculate the required inductor value with the following formulas:

Givens:

The LED I happened to have gives full brightness at 3.6 volts and 65 ma, so we will design the booster to nominally supply those values with a fresh, 1.5 volt battery. The switch saturation voltage will likely be less then 100 mV, but we'll use this as a worse case for the calculations.

Tcycle is the processor clock speed divided by the size of the PWM counter, 256 bits. To get a reasonably fast cycle time, the processor speed is set to 4.8 MHz, and the PWM counter is using this frequency without any prescaling. (A 9.6 MHz clock could be used, resulting in smaller inductors, but with slightly higher CPU current draw.) Therefore the cycle time is 4,800,000 Hz / 256 = 18,750 Hz or 53.3 us. So, our starting values for the calculations will be: Vin = 1.5 V, Vout = 3.6 V, Iout = 65 ma, Vsat = 0.1V, Vfwd = 0.3 V, tcycle = 53 us.

The first calculation is to determine the on/off duty cycle time
ton/toff = (Vout + Vfwd – Vin) / (Vin – Vsat) = (3.6 + 0.3 – 1.5) / (1.5 - .1) = 1.7

Now we need to know the on time to get the required voltage, which we find by first calculating the off time and subtract it from the cycle time:

toff = tcycle / (ton/toff + 1) = 53 us / (1.7+1) = 19 us (rounded up)

ton = tcycle – toff = 53 – 19 = 34 us

Now we can calculate the peak inductor current:

Ipk = 2 Iout (ton/toff+1) = 2 (.1A) (1.7+1) = 540 ma

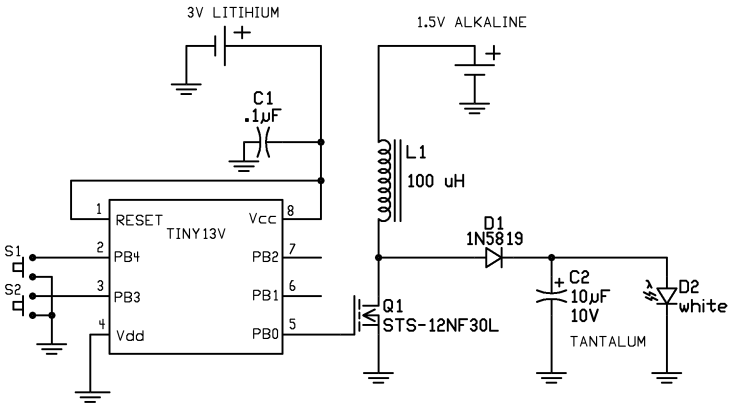
From this we can finally calculate the minimum inductor value needed:

Lmin = [(vin-Vsat)/Ipeak] ton = [(1.5-0.1)/.540]30 us = 77.7 uH

Unfortunately, 77 uH isn't a standard value, the closest being 100 uH and is what should be used.

As can be seen from the above calculations, boosting to a higher voltage results in a pretty high peak current from the low voltage supply. Average current is of course much lower integrated over the charging time of the inductor and the off time. Still, the battery needs to be able to supply that peak current.

Circuit for the booster is shown to the right. Q1 is in a SO-8 package and if SMT parts are used for the rest of the components, the circuit can be made reasonably small. Currently, S1 has no function. Pushing S2 will cycle from off to low beam to high beam to off again. A super cap could be added across the 1.5V battery to reduce peak current demand from the battery, but this would add some bulk to the final package.



Step down regulator:

Although the step up boost regulator allows using a single 1.5V battery to power the LED, battery life suffers. Even if the conversion was 100% efficient, (which it isn't by a long shot) the average current out of the battery has to be twice that drawn by the load. (since in this case we are basically doubling the voltage). This is because power must be conserved. If 300 mw is being drawn by the load, 300 mw has to be delivered by the source. Since watts = voltage x current, 300 mw at 3 V is 100 ma but is 200 ma at 1.5 V.

Much better battery life can be obtained by using a step down (buck) regulator than a step up (boost) regulator. By the same law of conservation of energy, if the load is operating at a voltage half that of the supply, in theory the supply only has to deliver half as much current as the load draws to use the same amount of power.

In practice, the traditional way of producing a lower voltage from a higher one is with the use of a dropping resistor. This could be an actual resistor or a transistor acting as one. Either way, power is wasted in the dropping process. The current from the source is equal to the current in the load because this is a series circuit. Therefore, if dropping the voltage in half, the source is using twice the power as the load with half being dissipated in the dropping resistor.

This waste of energy can be significantly improved on by using a switching regulator. A step down switching regulator uses an inductor in series with the load and a series connected switch. The inductor limits the current flowing into the load from the source when the switch is on, then a diode is used to dump the energy stored in the inductor back into the load when the switch turns off. Efficiencies of over 90% can be obtained this way. So, lets design a step down regulator. The same number of parts are used but in a different configuration than the step up version and the N channel MOSFET changes to a P channel type as now it is a series switch instead of

a shunt switch. The formulas for calculating the inductor value needed are also very similar to the ones used for the boost circuit.

Lets start with these givens: Vin = 6V, Vout = 3.6 V, Iout = 60 ma, Vsat = .1V, Vf = .3V, tcycle = 53 us.

ton/toff = (Vout + Vf) / (Vin – Vsat – Vout) = (3.6 + .3) / (6 - .1 – 3.6) = 1.7

toff = tcycle / (ton/toff) +1 = 53 / (1.7 + 1) = 19.7 us

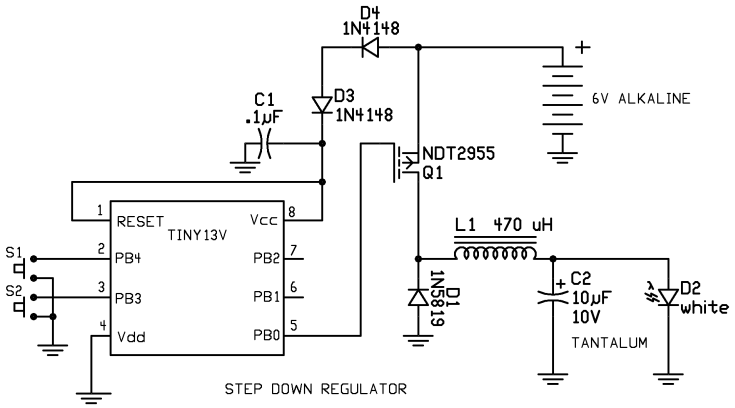
ton = tcycle – toff = 53 – 22 = 33.3 us

Ipk = 2 Iout = 2 (60 ma) = 120 ma

Lmin = [(Vin – Vsat – Vout) / Ipk] ton = [(6 - .1 – 3.6) / .12] 33.3 = 638 uH

680 uH is the closest standard value larger than 638, so is what is used.

The circuit for the step down regulator is to the left. Since we now have 6 volts to work with, a separate battery for the processor isn't needed. However, since the Tiny13 has a maximum input voltage rating of 6 volts and four fresh alkaline batteries will exceed this, so two silicon diodes are used in series with the processor supply pin to drop the voltage without adding any additional current. The schematic shows a 470 uH inductor, which was the result of using slightly different output voltage and current values in my initial calculations.



Practice vs Theory: The Tiny13 clock is running a little slow in my test circuit, so the cycle time is 60 us instead of the 53 used for the calculations. The on time is the same at 34 us, but the ratio of on/off time is off a little.

For the boost version, I measure a LED voltage of 3.7 V at 80 ma. Input current from the 1.5V supply peaks at about 400 ma and averages about 220 ma, which pretty much agrees with what we calculated, given the differences in timing and measurement accuracy.

For the buck version, I measure a LED voltage of 3.5 V at 50 ma. Input current peaks at 100 ma and averages about 30 ma. The average current is lower than the load current because some of the 100 ma peak current is restored to the load when there is no current being drawn during the off time.

We can now clearly see that while using a single AA battery for a LED power source can be done, the extra bulk and weight of using 4 batteries gives much better battery life resulting in the use of less batteries in the long term.

The firmware source code:

The source code for the Tiny13 is listed on the next two pages. This can be cut and pasted into the AVR Studio 4 development suite and compiled to generate the hex code for programming the processor. The program is nearly identical for both the step down and step up regulator. The difference is in the step down regulator the output of the PWM is normally low, while in the step up it is normally high. Which version to be used is determined by removing the semi-colon (which makes the assembler ignore that line) at the beginning of the line for the desired mode instruction. The same PWM timing is used for both versions as the on/off time is about the same. This makes sense since as in one case we are doubling the voltage in the other dividing it in half, so the same ratios apply. Additional brightness settings could be added to the program and a strobe or blinking feature added using the watch dog timer, used as a delay counter.

Below is the source code needed for the Tiny13 processor. It can be cut and pasted into AVR Studio 4 and complied to generate the hex file needed for programming the chip. Be sure to remove the semi-colon and text in front of instruction for desired step up or step down operating mode. These changes are needed in three places.

```
;PWM control for boost or buck switching voltage conversion.
;push button low beam, high beam and off control on PB3
;MOSFET drive on PB0
;PWM on time controlled by word in OCR0A, $FF = 0 on time.
;set fuses to 4.8 MHz RC clock, no clock divide

.include "tn13def.inc"

.def            temp = r16
.def            pwmword = r17
.def            pwrreg  = r18

.org            0000
                rjmp    reset           ;interrupt vectors
                nop
                clr     temp            ;pin change interrupt
                out     pcmsk,temp      ;turn off pin change interrupt so it doesn't retrigger on switch release
                rjmp    pinchange

;*****************************************************************************************************************
;start of program
;*****************************************************************************************************************
reset:          ldi     temp,low(ramend)  ;stack pointer address
                out     spl,temp          ;load stack pointer
                ldi     temp,$07          ;port directions, pb0,1,2 out, 3,4 in
                out     ddrb,temp         ;load port direction reg
; use for step up       ldi     temp,$18          ;initial port state pb0,1,2 low, 3,4 high
;use for step down      ldi     temp,$19          ;initial port state pb0, 3,4 high, 1, 2 low
                out     portb,temp        ;output values to port
                clr     pwrreg            ;clear register, indicates operating state
                sbi     acsr,acd          ;turn off analog comparator to save power
                ldi     temp,$08          ;value to enable pb3 pin change
                out     pcmsk,temp        ;output to pin change enable mask
                ldi     temp,$20          ;value to turn on pin change interrupts
                out     gimsk,temp        ;load general interrupt mask
                sei                       ;enable general interrupts
                ldi     temp,$30          ; set sleep mode to power down
                out     mcucr,temp        ;output to register
                sleep                     ;go to sleep, wait for switch interrupt to wake up
                nop                       ;program returns here after pin change interrupt
                ldi     temp,$08          ;turn pin change interrupt back on
                out     pcmsk,temp

gosleep:        ldi     temp,$20          ;value to set idle sleep mode, allow timers to run
                out     mcucr,temp        ;load control register
                sleep                     ;go to idle sleep mode, wait for switch interrupt
                nop                       ;return from interrupt, do nothing
                ldi     temp,$08
                out     pcmsk,temp        ;turn pin change interrupt back on
                brtc    gosleep           ;test for off mode next step, if not go to idle sleep mode
                ldi     temp,$30          ;turn off selected, so set sleep mode to power down
                out     mcucr,temp        ;output to register
                sleep                     ;go to sleep, power down mode
                nop                       ;woke up from off mode
                ldi     temp,$08          ;turn pin change interrupt back on
                out     pcmsk,temp
                clt                       ;clear t flag, led is now on
                rjmp    gosleep           ;go to idle mode sleep now that led in on

pinchange:      tst     pwrreg            ;test if register is cleared
                brne    highbeam          ;pwrreg is not cleared so go to high beam settings
                inc     pwrreg            ;increment reg to indicate low beam mode on
lowbeam:        ldi     temp,$a3          ;compare value for low beam PWM time
                out     ocr0a,temp        ;output to timer 0 compare register
;use for step up        ldi     temp,$c3          ;set PWM modes to fast mode, enable output compare on portb,0, inverted
;use for step down      ldi     temp,$83
                out     tccr0a,temp       ;output values to control register
                ldi     temp,$01          ;set prescaller counter to direct output, no prescalling
```

```
                out     tccr0b,temp     ;output to register
                rcall   debounce        ;go to switch debounce routine
                reti                    ;return from interrupt

highbeam:       sbrs    pwrreg,0        ;test if high beam already on
                rjmp    off             ;yes it was, so go turn off
                inc     pwrreg          ;increment register to clear 0 bit, but leave none zero value in register
                ldi     temp,$73        ;high beam PWM value
                out     ocr0a,temp      ;output to compare register
                rcall   debounce        ;debounce switch
                reti                    ;return from interrupt

off:            clr     temp            ;clear temp register
                clr     pwrreg          ;clear pwr register to indicate off mode turned on
                out     tccr0a,temp     ;clear timer 0 register, clear PWM mode
                out     tccr0b,temp     ;clear timer 0 register, turn off counter
; use for step up       cbi     portb,0         ;set MOSFET drive port low
;use for step down      sbi     portb,0         ;set MOSFET drive port high
                set
                rcall   debounce        ;debounce switch
                reti

debounce:       ldi     r22,$0f
dbj0:           sbis    pinb,3          ;long switch debounce timer
                rjmp    dbj0
dbj1:           dec             r20
                brne    dbj1
                dec             r21
                brne    dbj1
                dec             r22
                brne    dbj1
                ret
```