ENGR 498: Design for the Internet of Things – Energy and Sleep

Dr. Jason Forsyth
Department of Engineering
James Madison University

General CPU Performance

- In any computer system to calculate the amount of *time* required by some tasks you need to know the number of *instructions* required to complete the task and how quickly those instructions can be executed.
- Each *instruction* take a certain number of CPU *cycles* to complete. This is called *cycles per instruction* (CPI) where each cycle is one oscillation of the processor's clock (f).
- Time = $Cycles \ * Cycle\ Period = instr.* CPI \ * period = \frac{instr.*CPI}{f}$

General CPU Performance

- Simple instructions (move, add, shift, subtract…) may execute in a single cycle (CPI = 1)
- Complex instructions (multiple, divide, modulus) may execute over several cycles (CPI > 1).
- How you write your code influences which instructions are implemented.
  - int a * int b << float a * float b

32. Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
EIJMP		Extended Indirect Jump to (Z)	PC ←(EIND:Z)	None	2
JMP	k	Direct Jump	PC ← k	None	3
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	4
ICALL		Indirect Call to (Z)	PC ← Z	None	4
EICALL		Extended Indirect Call to (Z)	PC ←(EIND:Z)	None	4
CALL	k	Direct Subroutine Call	PC ← k	None	5
RET		Subroutine Return	PC ← STACK	None	5
RETI		Interrupt Return	PC ← STACK	I	5
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3

The more time you spend on tasks the more energy you require.

CPU Power

- CPU power is combination of Dynamic and Static power

- $Power = P_{dynamic} + P_{short} + P_{leak}$
- $Power = \alpha C V^2 f + \tau \alpha V I_{short} f + I_{leak} V$

- Leakage current: due to quantum tunneling effects in transistor
- Short current: temporary connection between VDD and GND on transistor switch

CPU Energy

- Power is useful when selecting a supply, but energy is more important when consider fixed charge sources. Recall that $t = \frac{cycles}{f}$ for some operation.

- $E = P * t = (\alpha C V^2 f + \tau \alpha V I_{short} f + I_{leak} V) * \frac{cycles}{f}$

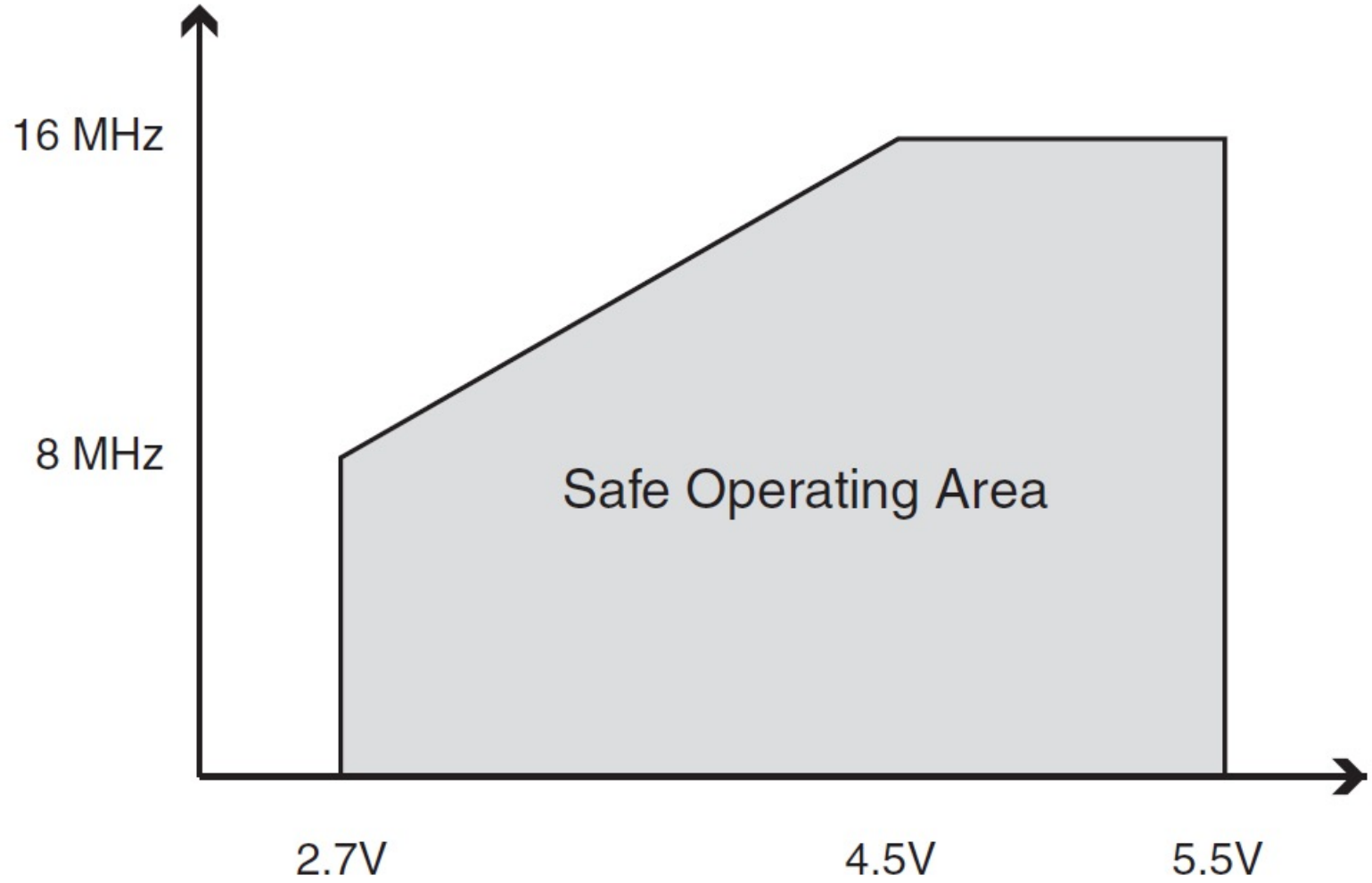
- $E = P * t = \left(\alpha C V^2 + \tau \alpha V I_{short} + \frac{I_{leak} V}{f}\right) * cycles$

Voltage and Frequency Scaling

- To reduce power the processor can *lower voltage* or *lower frequency*.
- If the voltage is “too low” then the processor cannot execute at a given frequency. Glitches will occur as the chip cannot switch fast enough.
- The frequency can always be reduced at a given voltage.

Voltage and Frequency Scaling

Figure 29-2. Maximum Frequency vs. V_{CC}



Effect of Voltage and Frequency Scaling on Power

- Modern processors support scaling of frequency and voltage to support better energy efficiencies. Assuming $f \alpha V$ by some factor of S:

- $Power = \alpha C(Vs)^2(fs) + \tau\alpha(Vs)I_{short}(fs) + I_{leak}(Vs)$
- $= \alpha CV^2fs^3 + \tau\alpha VI_{short}fs^2 + I_{leak}Vs$

- Significant reduction in dynamic power, but less in static power
  - If voltage and frequency are cut in half (1/2) then dynamic power is reduced to (1/8).

Effect of Voltage and Frequency Scaling on Energy

- $E = P * t = \left(\alpha C V^2 + \tau \alpha V I_{short} + \frac{I_{leak} V}{f}\right) * cycles$
- $= \left(\alpha C V^2 s^2 + \tau \alpha V I_{short} s + \frac{I_{leak} V}{f}\right) * cycles$

- Reduction in dynamic and short circuit energy, however no reduction in leakage energy

- Leakage actually increase as f decreases. Makes sense as it takes more time to complete some operation

Run Fast or Slow Down?

- Assume a task has a deadline of 100ms but only requires 50ms to complete. What is the most efficient choice?
  - Option A: run for full 50ms, then power off for remaining 50ms
  - Option B: reduce F and V by 2x to take all 100ms

- Which do you choose?

Sleep and Power Example

- Assume we have a task that has a deadline of 100ms but we can accomplish it at 50ms at V_{max} and f_{max}

- Option #1: Run for 50ms, then sleep for the remaining 50ms
  - $E = P * \frac{t}{2} = \alpha C V^2 f \ * instructions$
  - $E = \left(\frac{1}{2}\right) \alpha C V^2 f$ * instructions

- Option #2: Reduce f and V to take all 100ms
  - $E = P * t = \alpha C \left(\frac{V}{2}\right)^2 \left(\frac{f}{2}\right) * instructions = \left(\frac{1}{8}\right) \alpha C V^2 f * instructions$

- Option #2 is better even though it takes longer

Run Fast or Slow Down?

Static power/energy was missing from the previous calculation.

Will have a tremendous effect on the efficiency of the processor…

Sometimes it’s better to just go to sleep.

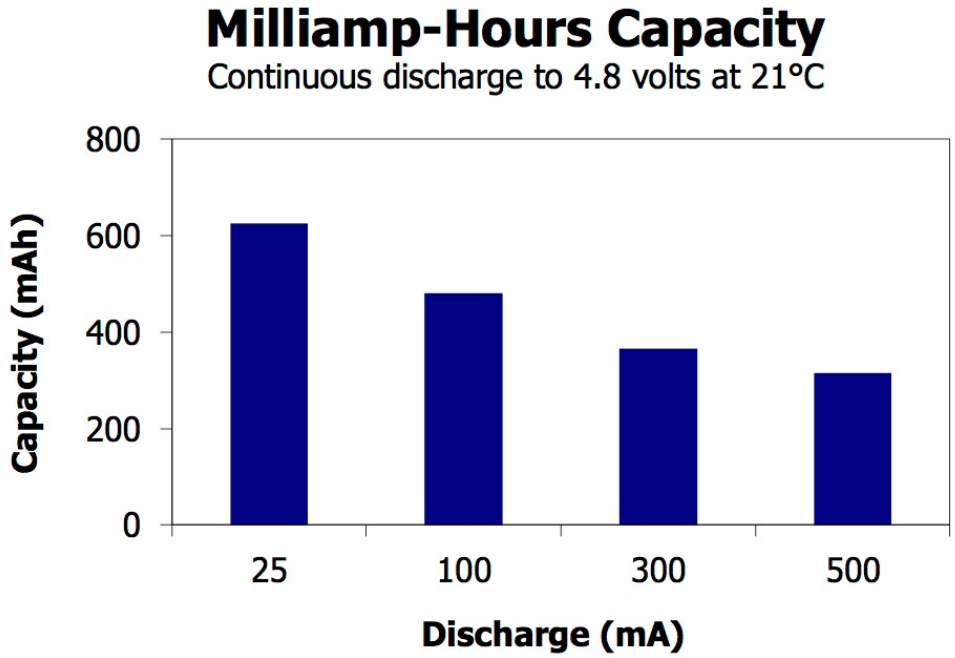
Energy Impact of Other Peripherals

- Previous calculations only considered CPU. However any peripheral may consume power. As a better model, consider System Power
- System Power = $a_2 s^3 + a_1 s + a_0$
- a_2: things that scale with voltage and frequency (CPU)
- a_1: things that scale with frequency (RAM, Flash) or voltage (EEPROM, Accelerometer…etc.). Items with low-power mode.
- a_0: things that don’t scale or no low-power mode

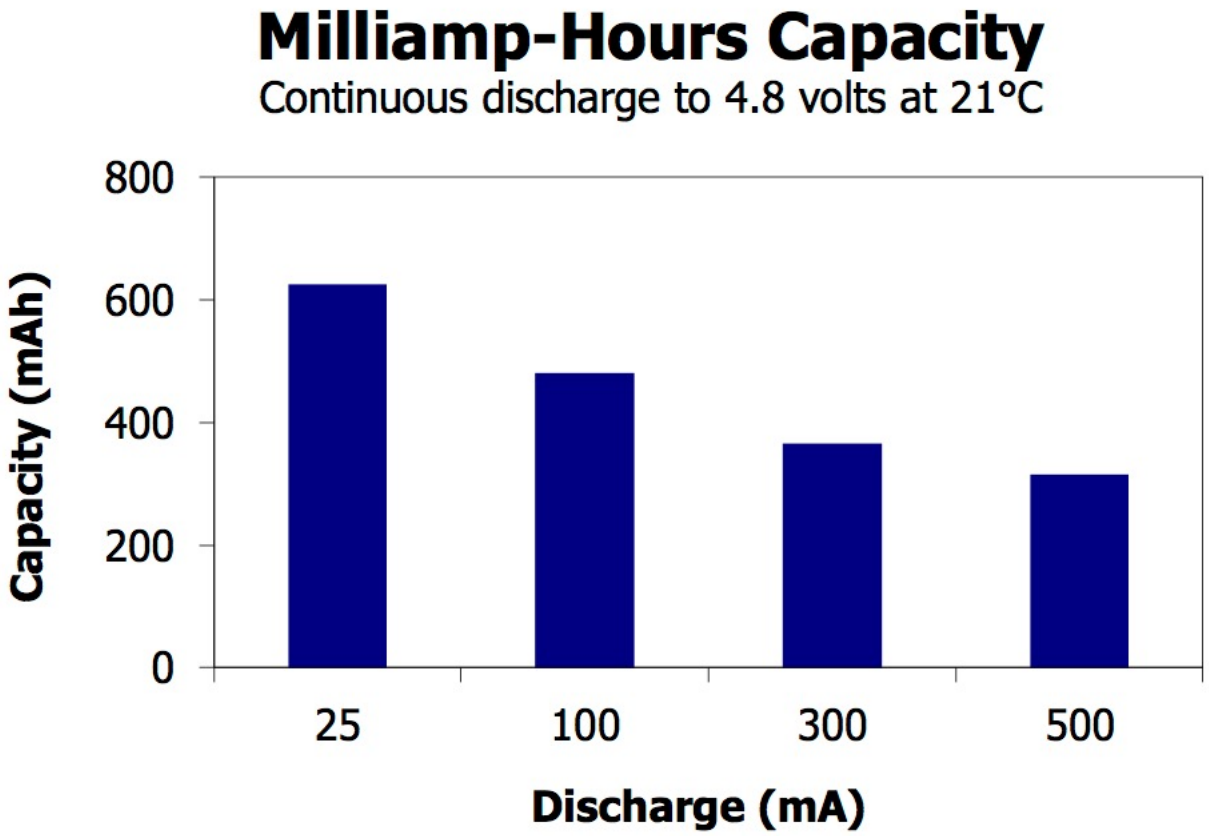
Consider the limited resources powering your system

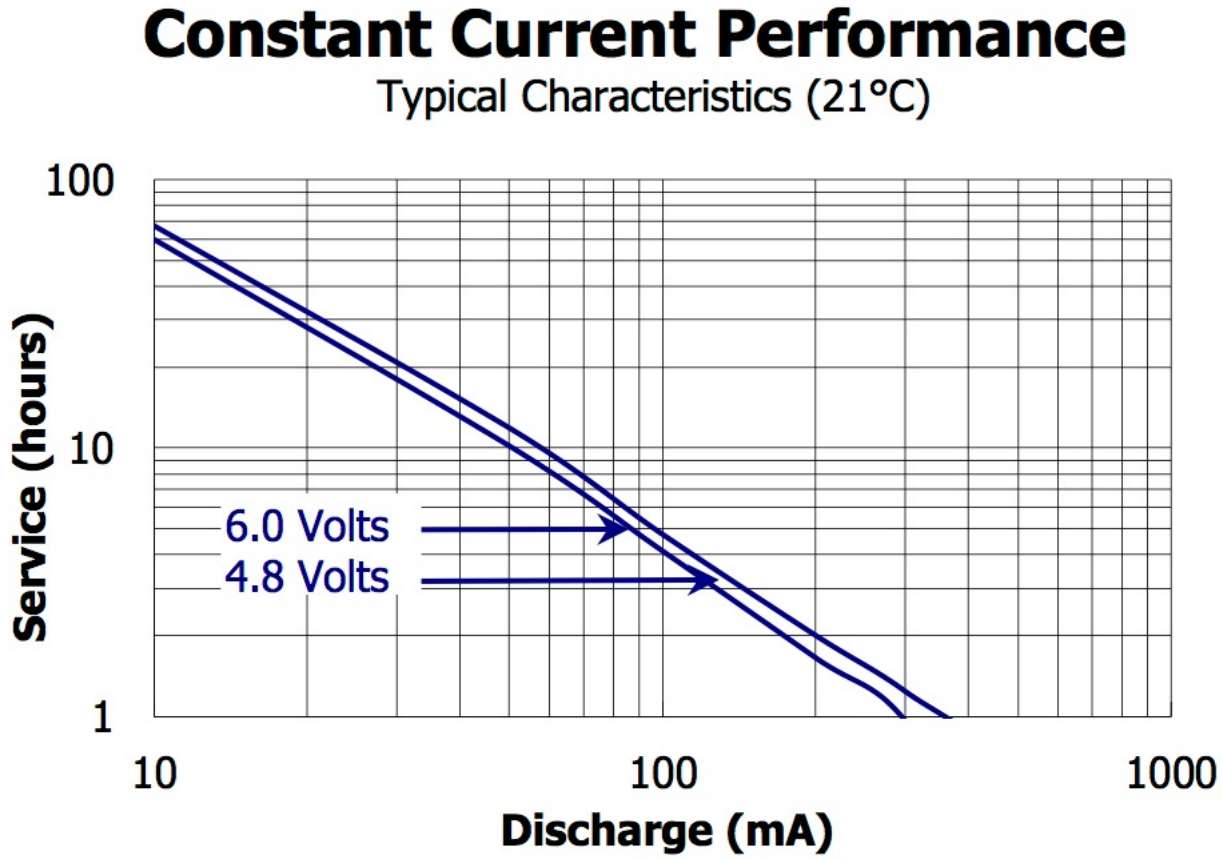
System Lifetime on Batteries

- Real embedded systems aren't connected to a power supply.
- Battery lifetime are dependent upon *charge capacity*. Generally measured in milliamp-hours (mAh). Effects are non-linear.



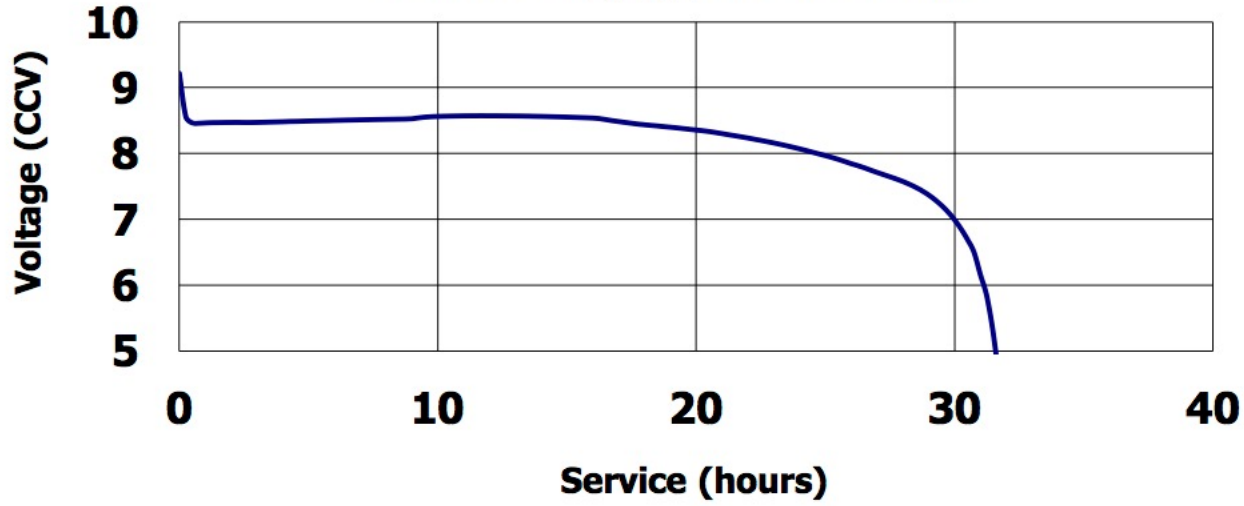
Alkaline


Milliamp-Hours Capacity
Continuous discharge to 4.8 volts at 21°C
800
600
400
200
0
Capacity (mAh)
25
100
300
500
Discharge (mA)

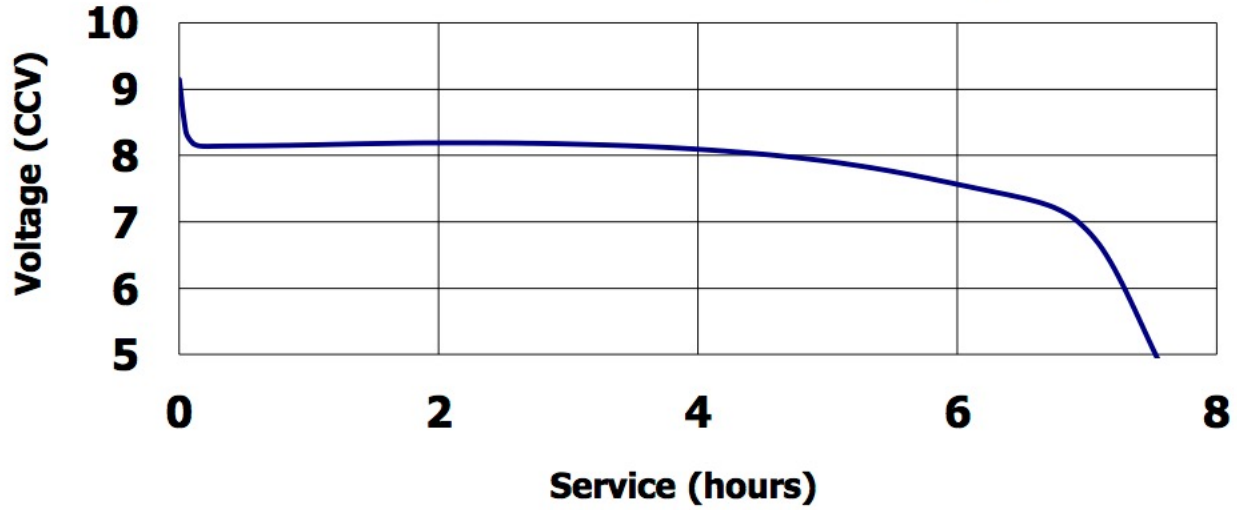

Constant Current Performance
Typical Characteristics (21°C)
100
10
1
Service (hours)
6.0 Volts
4.8 Volts
10
100
1000
Discharge (mA)

Typical Discharge Performance (21°C)

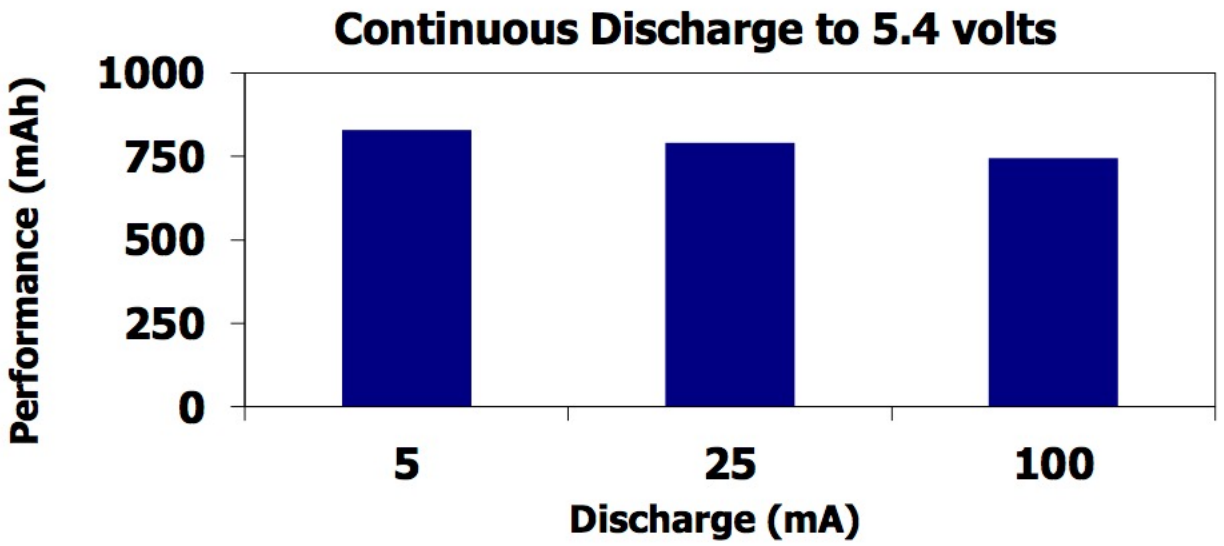
25mA Continuous Discharge

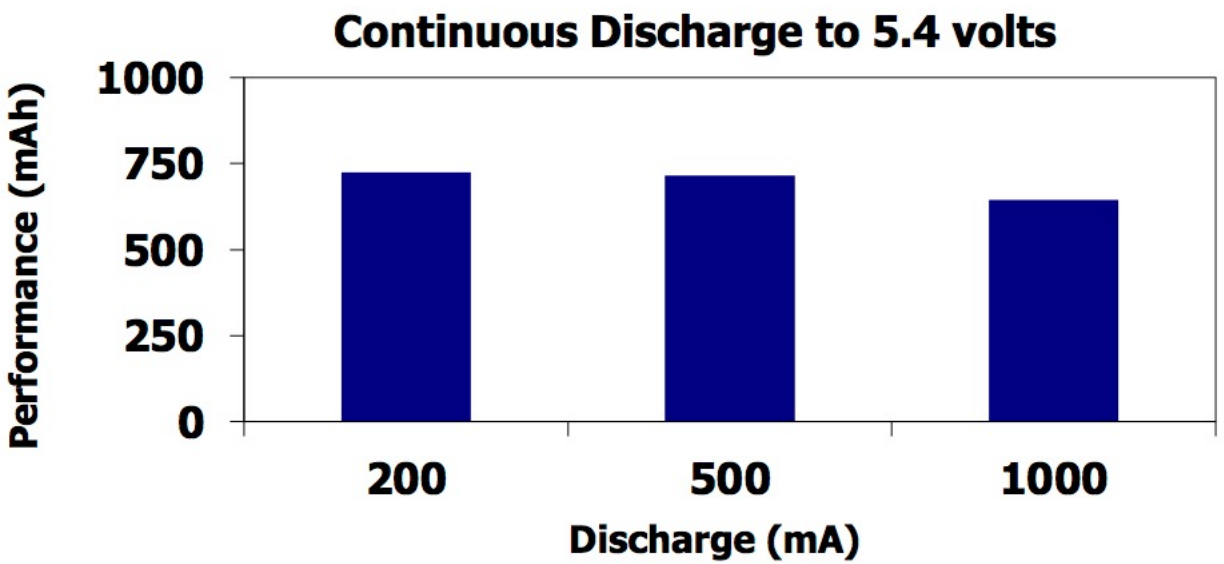


100 mA Continuous Discharge

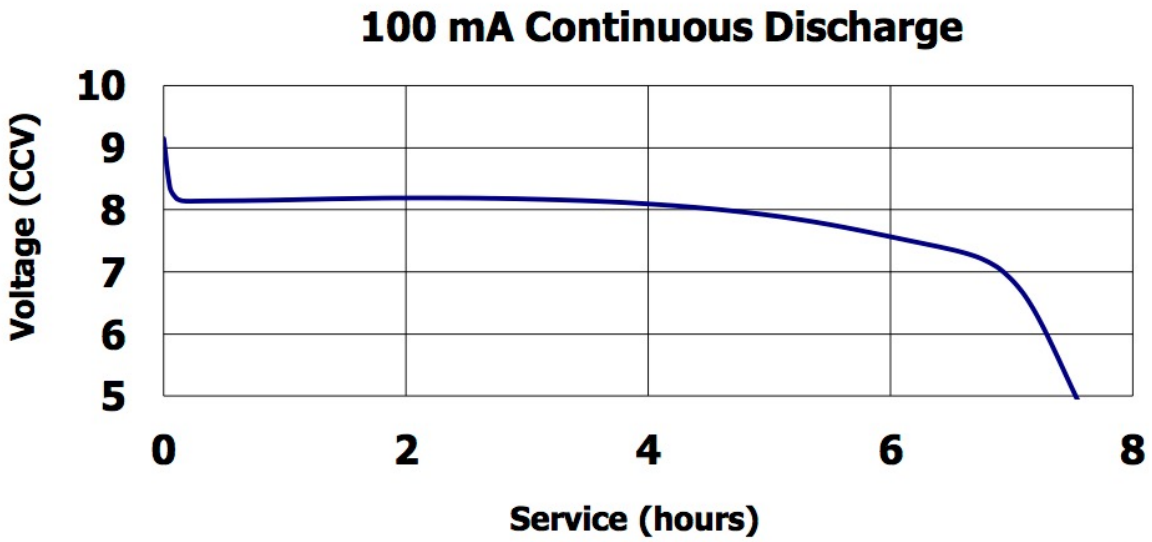


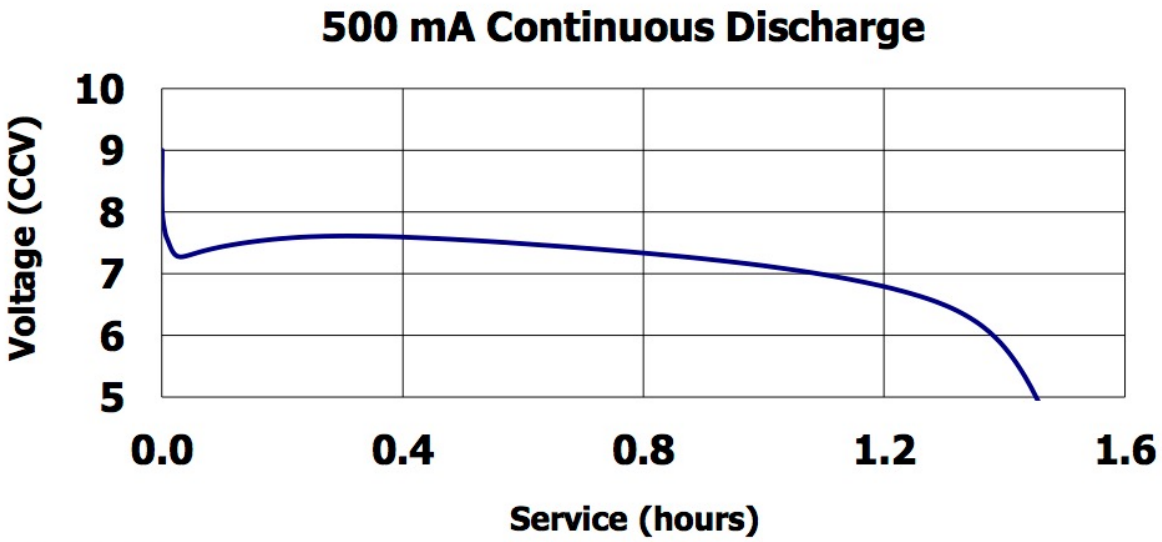
Lithium


Continuous Discharge to 5.4 volts
Performance (mAh)
1000
750
500
250
0
5
25
100
Discharge (mA)


Continuous Discharge to 5.4 volts
Performance (mAh)
1000
750
500
250
0
200
500
1000
Discharge (mA)

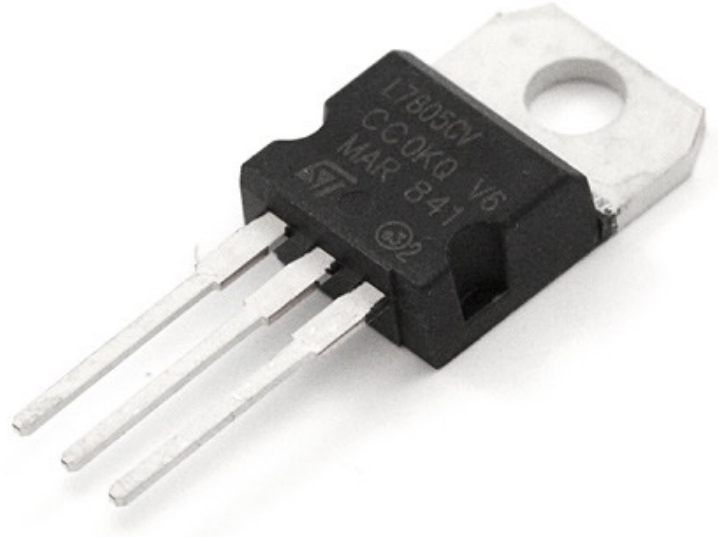
Lithium


100 mA Continuous Discharge
Voltage (CCV)
10
9
8
7
6
5
0
2
4
6
8
Service (hours)


500 mA Continuous Discharge
Voltage (CCV)
10
9
8
7
6
5
0.0
0.4
0.8
1.2
1.6
Service (hours)

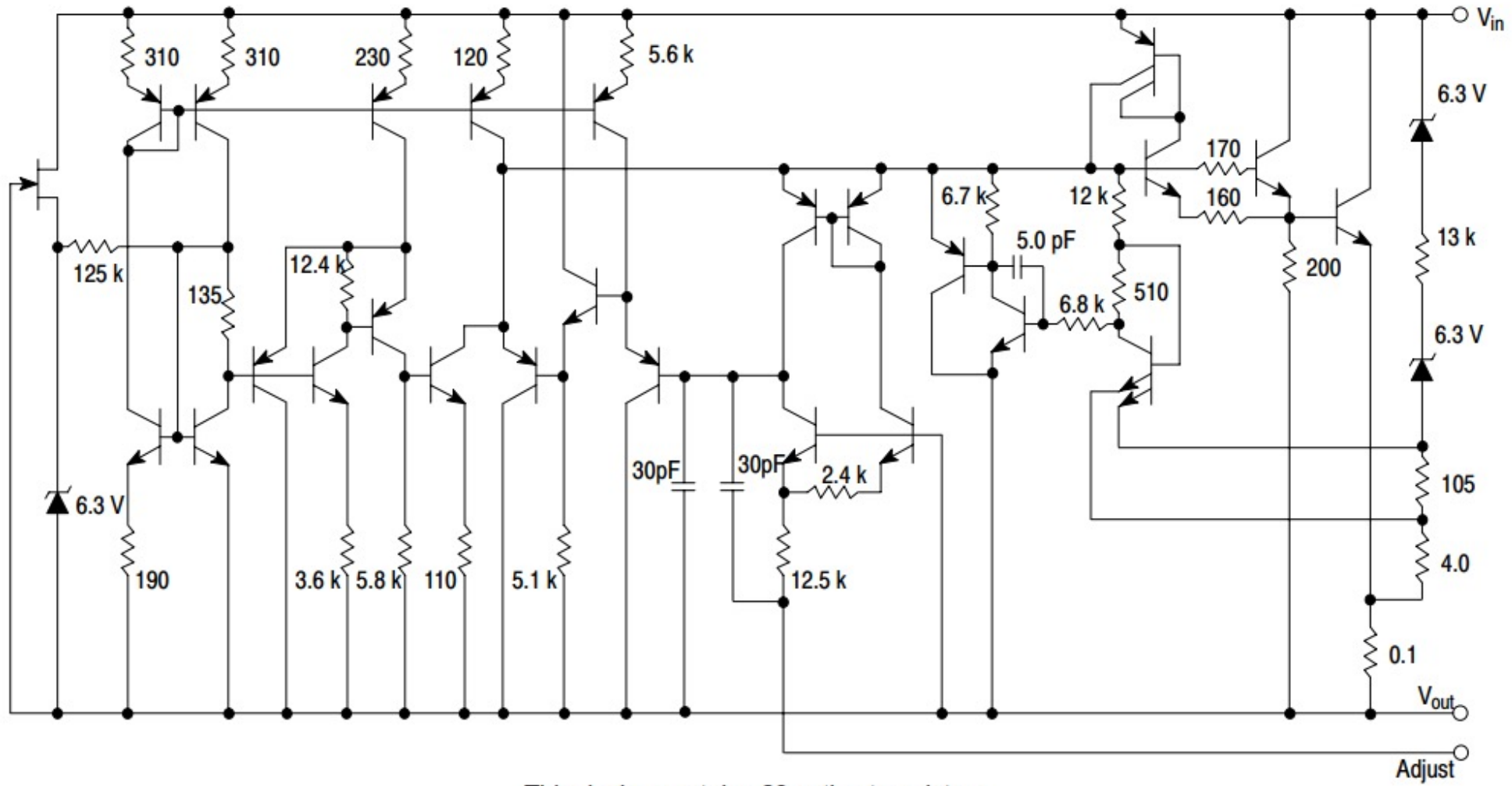
Battery Lifetime Equations

- $L_{linear} = \alpha * \frac{Cap}{i_{avg}}$
- $L_{switching} = \frac{V_{source} * Cap}{V_{system} * i_{avg}}$
- $\alpha \approx 0.8$ for linear regulators



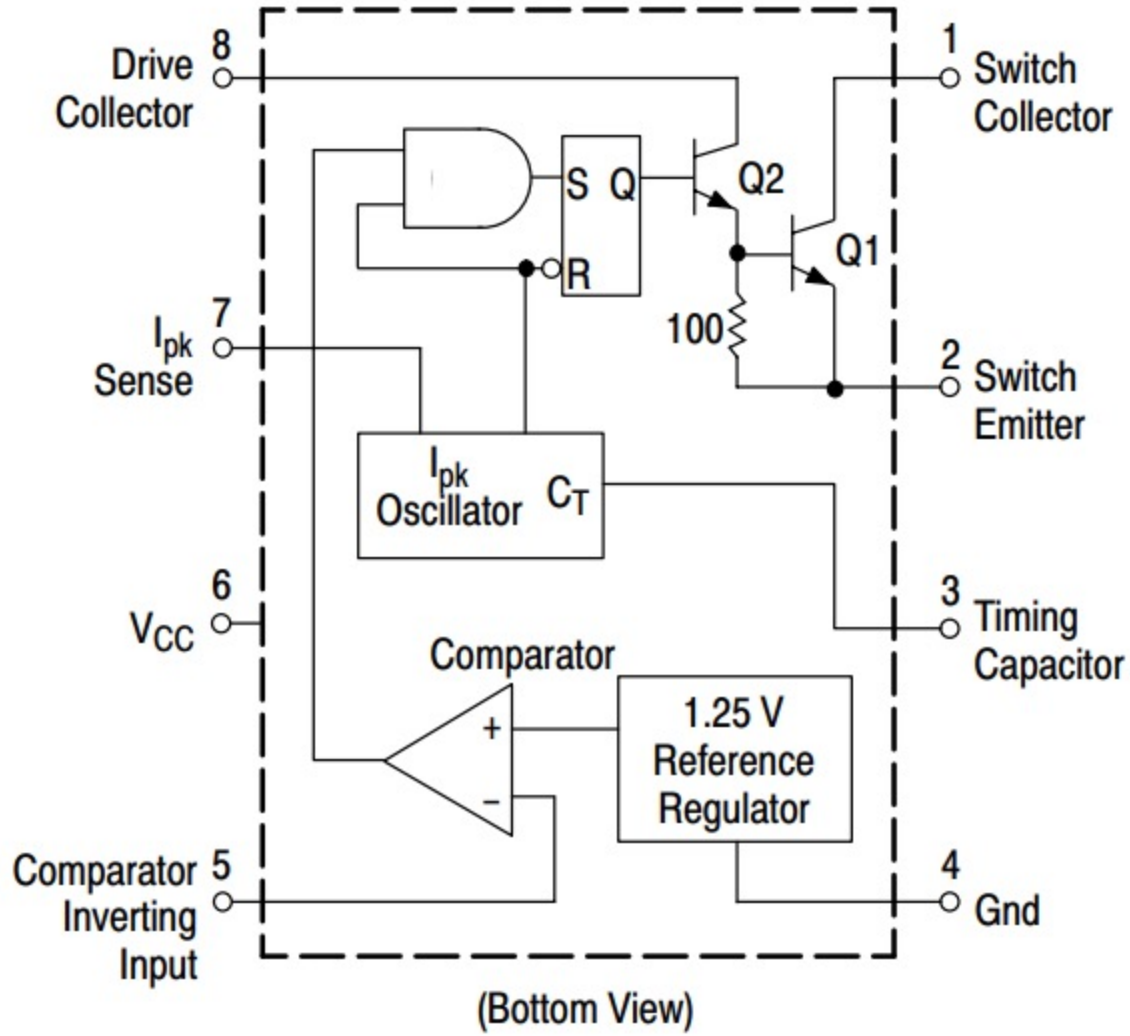
Linear Regulator

LM317, NCV317



This device contains 29 active transistors.

Switching Regulator



The best way to reduce power is to turn things off

Amtel Sleep Modes

Chapter 7 – Atmel 32u4 Datasheet

Low-Power Operation and Sleep

- ATMega32u4 supports five sleep modes: idle, power down; power save; stand by; and extended standby

- Each sleep level disables different peripherals and can be woken from different sources

- Key questions: What are you waking from? And what peripherals can be turned off?

Table 7-1. Active Clock Domains and Wake-up Sources in the Different Sleep Modes

	Active Clock Domains				Oscillators	Wake-up Sources							
Sleep Mode	**clk_{CPU}**	**clk_{FLASH}**	**clk_{IO}**	**clk_{ADC}**	**Main Clock Source Enabled**	**INT6, INT3:0 and Pin Change**	**TWI Address Match**	**SPM/ EEPROM Ready**	**ADC**	**WDT Interrupt**	**Other I/O**	**USB Synchronous Interrupts**	**USB Asynchronous Interrupts(3)**
Idle			X	X	X	X	X	X	X	X	X	X	X
ADCNRM				X	X	X(2)	X	X	X	X		X	X
Power-down						X(2)	X			X			X
Power-save						X(2)	X			X			X
Standby(1)					X	X(2)	X			X			X
Extended Standby					X	X(2)	X			X			X

Notes:
1. Only recommended with external crystal or resonator selected as clock source.
2. For INT6, only level interrupt.
3. Asynchronous USB interrupts are VBUSTI and WAKEUPI.

Wake-Up Sources

- External interrupts: level or edge triggered. Edge trigger requires an active clock
- Watch Dog Timer: can be to periodically expire and wake up
- I2C address match: if device is on I2C and listed as slave or multi-master
- In your project, what are you going to cause a wakeup?

Entering Sleep Mode

- Set sleep bits (SM2-SM0) in the SMCR register to select the mode
  - Power down, standby…etc.
- Enabled sleep with the SE bit in SMCR
- Execute a sleep instruction sleep_cpu();
  - Use <avr/sleep.h> header
  - http://www.microchip.com/webdoc/AVRLibcReferenceManual/group__avr__sleep.html

7.9.1 Sleep Mode Control Register – SMCR

The Sleep Mode Control Register contains control bits for power management.

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	SM2	SM1	SM0	SE	SMCR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 3, 2, 1 – SM2..0: Sleep Mode Select Bits 2, 1, and 0**

These bits select between the six available sleep modes as shown in Table 7-2.

Table 7-2. Sleep Mode Select

SM2	SM1	SM0	Sleep Mode
0	0	0	Idle
0	0	1	ADC Noise Reduction
0	1	0	Power-down
0	1	1	Power-save
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Standby(1)
1	1	1	Extended Standby(1)

Practical Guide to Sleep Mode

1. Set sleep mode
2. Clear interrupts to we execute atomically
3. If ready for sleep
    1. Enable the sleep bit
    2. Re-enable interrupts (so we can wake up)
    3. Sleep the CPU
    4. Disable the sleep bit (we've now woken up)
4. Enable interrupts

```
#include <avr/interrupt.h>
    #include <avr/sleep.h>

...
  set_sleep_mode(<mode>);
  cli();
  if (some_condition)
  {
    sleep_enable();
    sei();
    sleep_cpu();
    sleep_disable();
  }
  sei();
```

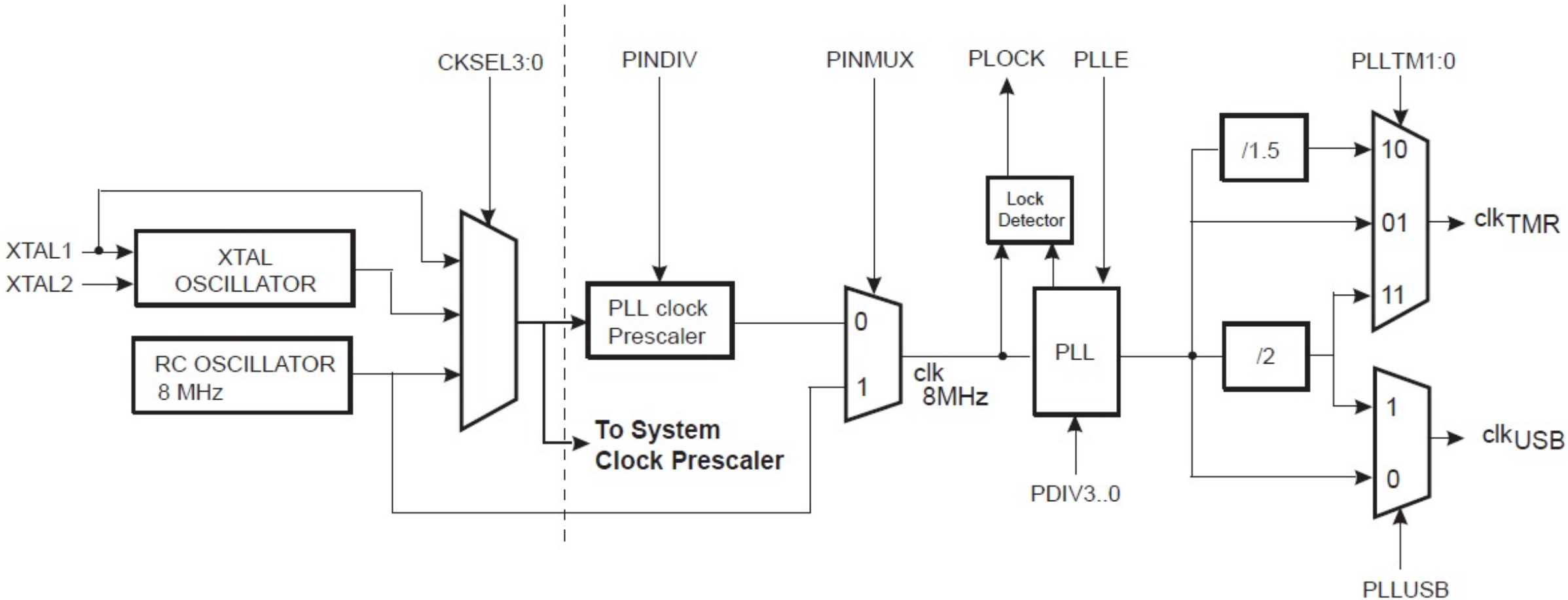
Note: if interrupts are not enabled before you go to sleep then you will not wake up!

Further Power Reduction

- Recall that power consumption for a uC is:
- $P = P_{dyanmic} + P_{static}$
- $P = \left(\frac{1}{2}\right) CV^2 f + I_{leak} V + \tau \alpha V I_{short} f$

- If you disable the clock to a peripheral, then power reduction is significantly reduced.

PLL Clocking System



Power Reduction Register to Disable Clocks

7.9.2 Power Reduction Register 0 - PRR0

Bit	7	6	5	4	3	2	1	0	
	PRTWI	–	PRTIM0	–	PRTIM1	PRSPI	–	PRADC	PRR0
Read/Write	R/W	R	R/W	R	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 - PRTWI: Power Reduction TWI**

Writing a logic one to this bit shuts down the TWI by stopping the clock to the module. When waking up the TWI again, the TWI should be re initialized to ensure proper operation.

- **Bit 6 - Res: Reserved bit**

This bits is reserved and will always read as zero.

Can be used in active mode to reduce overall power as well. Turn off interfaces that you don’t need.

Symbol	Parameter	Condition	Min.[5]	Typ.	Max.[5]	Units
I_{IH}	Input Leakage Current I/O Pin	V_{CC} = 5.5V, pin high (absolute value)			1	µA
R_{RST}	Reset Pull-up Resistor		30		60	kΩ
R_{PU}	I/O Pin Pull-up Resistor		20		50	
I_{CC}	Power Supply Current[6]	Active 4MHz, V_{CC} = 3V (ATmega16U4/ATmega32U4)			5	mA
		Active 8MHz, V_{CC} = 5V (ATmega16U4/ATmega32U4)		10	15	
		Active 16MHz, V_{CC} = 5V (ATmega16U4/ATmega32U4)			27	
		Idle 4MHz, V_{CC} = 3V (ATmega16U4/ATmega32U4)			2	
		Idle 8MHz, V_{CC} = 5V (ATmega16U4/ATmega32U4)			6	
	Power-down mode	WDT enabled, V_{CC} = 3V, Regulator Disabled		<10	12	µA
		WDT disabled, V_{CC} = 3V,Regulator Disabled		1	5	

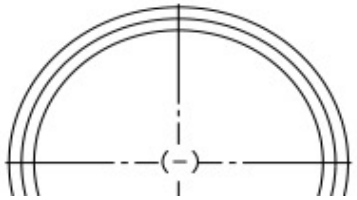
How much would you have to do to make your device last for one week?

ENERGIZER CR2032

Lithium Coin

Industry Standard Dimensions

mm (inches)



Specifications

Classification:	"Lithium Coin"
Chemical System:	Lithium / Manganese Dioxide (Li/MnO_2)
Designation:	ANSI / NEDA-5004LC, IEC-CR2032
Nominal Voltage:	3.0 Volts
Typical Capacity:	235 mAh (to 2.0 volts) (Rated at 15K ohms at 21°C)
Typical Weight:	3.0 grams (0.10 oz.)
Typical Volume:	1.0 cubic centimeters (0.06 cubic inch)
Max Rev Charge:	1 microampere
Energy Density:	198 milliwatt hr/g, 653 milliwatt hr/cc
Typical Li Content:	0.109 grams (0.0038 oz.)
Operating Temp:	-30C to 60C
Self Discharge:	~1% / year

Let's Assume the Current Systems

- If you ran continuously in *active* mode (5mA)
- $L_{switching} = \frac{V_{source} * Cap}{V_{system} * i_{avg}} = \frac{3.7V * 110mAh}{3.3V * 5mA} = 24h$
- If you needed a week (168h) then:
  - $i_{avg} = \frac{V_{source} * Cap}{V_{system} * L_{switch}} = \frac{3.7V * 110mAh}{3.3V * 168h} = 734\mu A$

The system would to sleep to achieve the lifetimes OR a different battery selected.

Summary

- Power is a significant issue for embedded system. If the device doesn’t last, who will use it?
- All components consume energy. Turn off things when not needed and slow down where possible.
- Use “sleep” to turn off while nothing “interesting” is happening. See the activity mode on the accelerometer.
- Will measure power consumption in later lab.