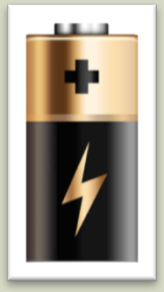
POWER SAVING USING MICROCONTROLLERS

Why and How?

INTRODUCTION

- Why is reducing power consumption important?
- Hardware vs. Software methods
- The need to balance power consumption with performance needs
- Examples

POWER FLOW IN AND OUT



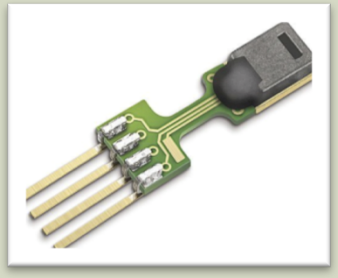
Vcc

MCU

Vss

Output pin

Input pin



Dissipated as heat

ENERGY CONSERVATION

- Vcc
- Input pin

=

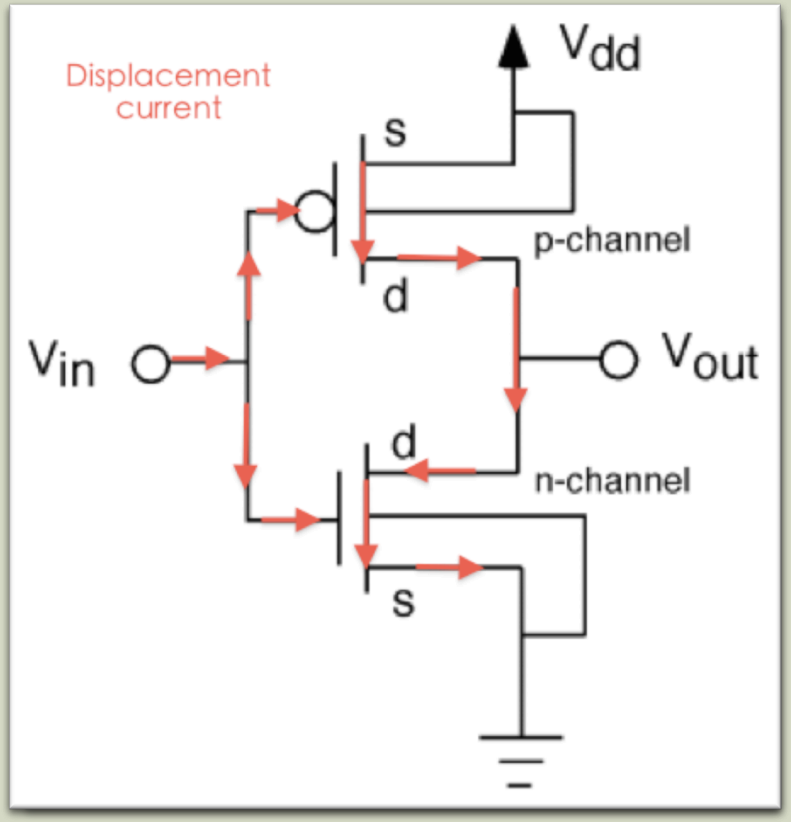
- Vss(ground)
- Dissipated as heat
- Output pin

Power Saving ?

Less current from Vcc and Input pin

Reduce the current to ground, heat and Output pin

SAVE POWER FROM GROUND AND OUTPUT



•CMOS inverter

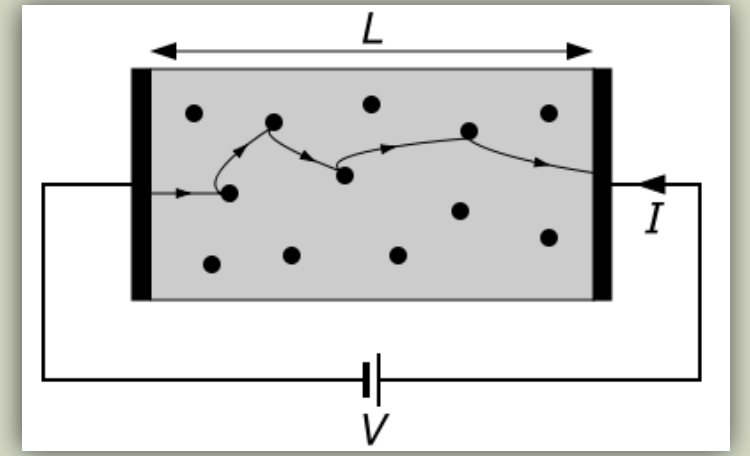
$$P = I_{DDQ} V_{DD} + Cout V_{DD}^2 f$$

Leakage current

Output capacitance

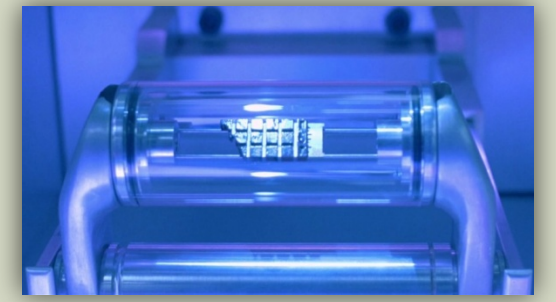
Hardware designer	User
Leakage current	Frequency
Vdd	
Output capacitance	

SAVE POWER FROM HEAT DISSIPATION



•In a conductor, heat was caused by electron collisions during drifting

•Superconductor



T800 Terminator

HOW TO SAVE POWER

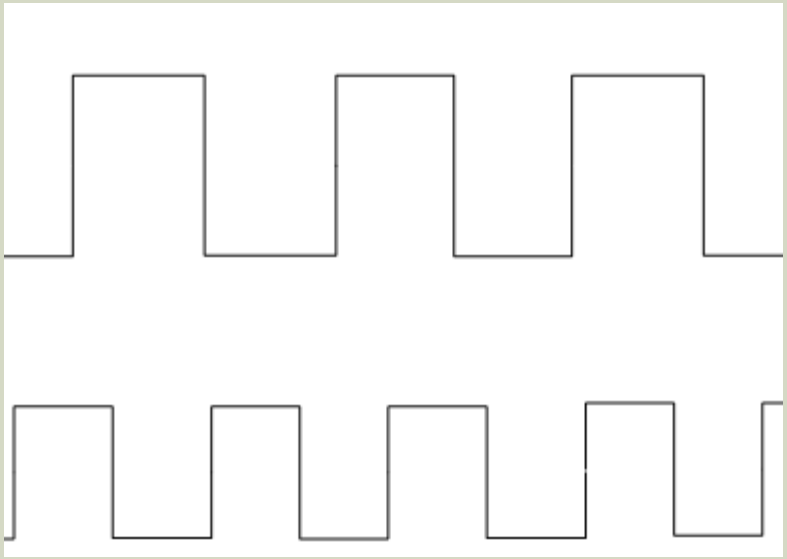
- Power Down Modes
- Clocking Systems
- Interrupts
- Peripherals

POWER-DOWN MODES

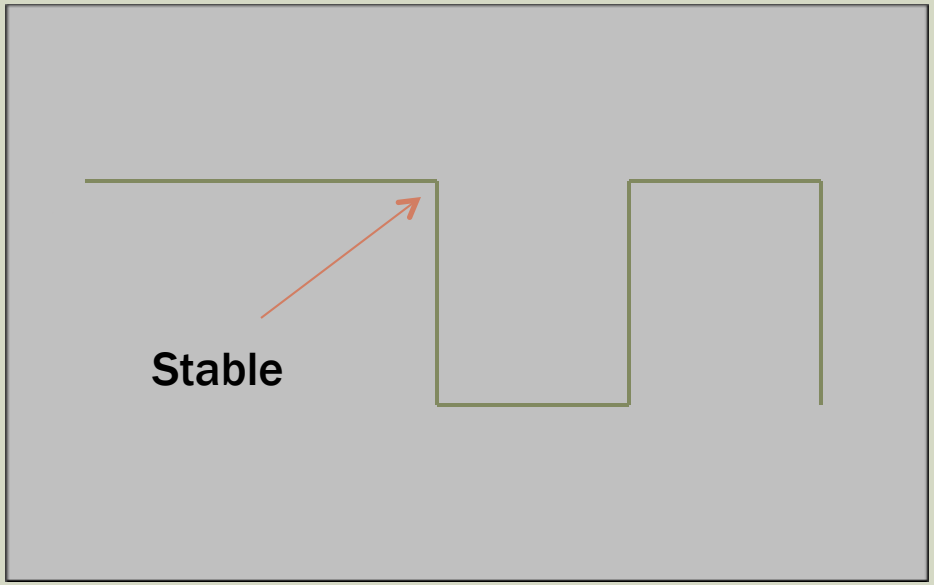
- Most important feature to meet current-budget
- Control clock frequency
- Control CPU power

CLOCKING SYSTEMS

- Microcontrollers may enter and exit low-power modes several times per second
- Current is wasted while CPU waits for stable clock
- Most low-power MCUs have “instant on clocks”



INSTANT-ON CLOCKS



- Some ready for CPU in less than 10-20us
- Some have two stage clock wake up
- Provides a low freq. clock while high freq. clock stabilizes
  - Can take 1ms longer
- On these devices, CPU may be operational in 15us
  - But runs on incorrect frequency
- CPU consumes less current at low frequencies
- Inaccurate frequency can relate to inaccurate timing
  - If accurate timing is necessary, the CPU must wait for the clock to stabilize.

INTERRUPTS

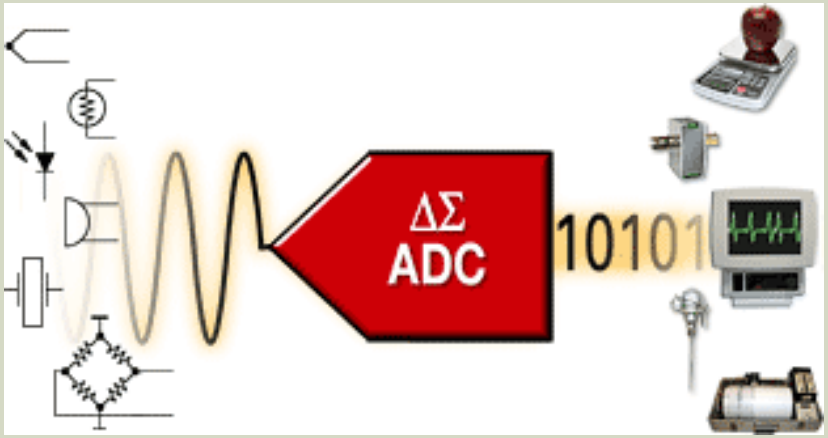
- Goes hand in hand with system clock flexibility
- Bring MCU out of low-power mode
  - More interrupts, more current-saving flexibility
- Without interrupt capability, MCU must poll the keypad or buttons often
  - Controlling polling requires a timer
    - Additional current
- With interrupt, CPU can sleep until a button is pushed

WHAT ARE MCU PERIPHERALS?

- Peripherals are components on a chip which have programmable input/ output capability

- Examples:
  - Timers
  - Event Counters
  - PWM generators
  - ADC/DAC

LOW POWER PERIPHERALS



- Some MCUs don't have low power peripherals
- Two types of low power peripheral ability
  - Individual enable/disable
  - Automatic enable/disable
- A true low-power peripheral is one that consumes no current when not in use

SPECIAL DETAILS OF POWER SAVING ON MCUS

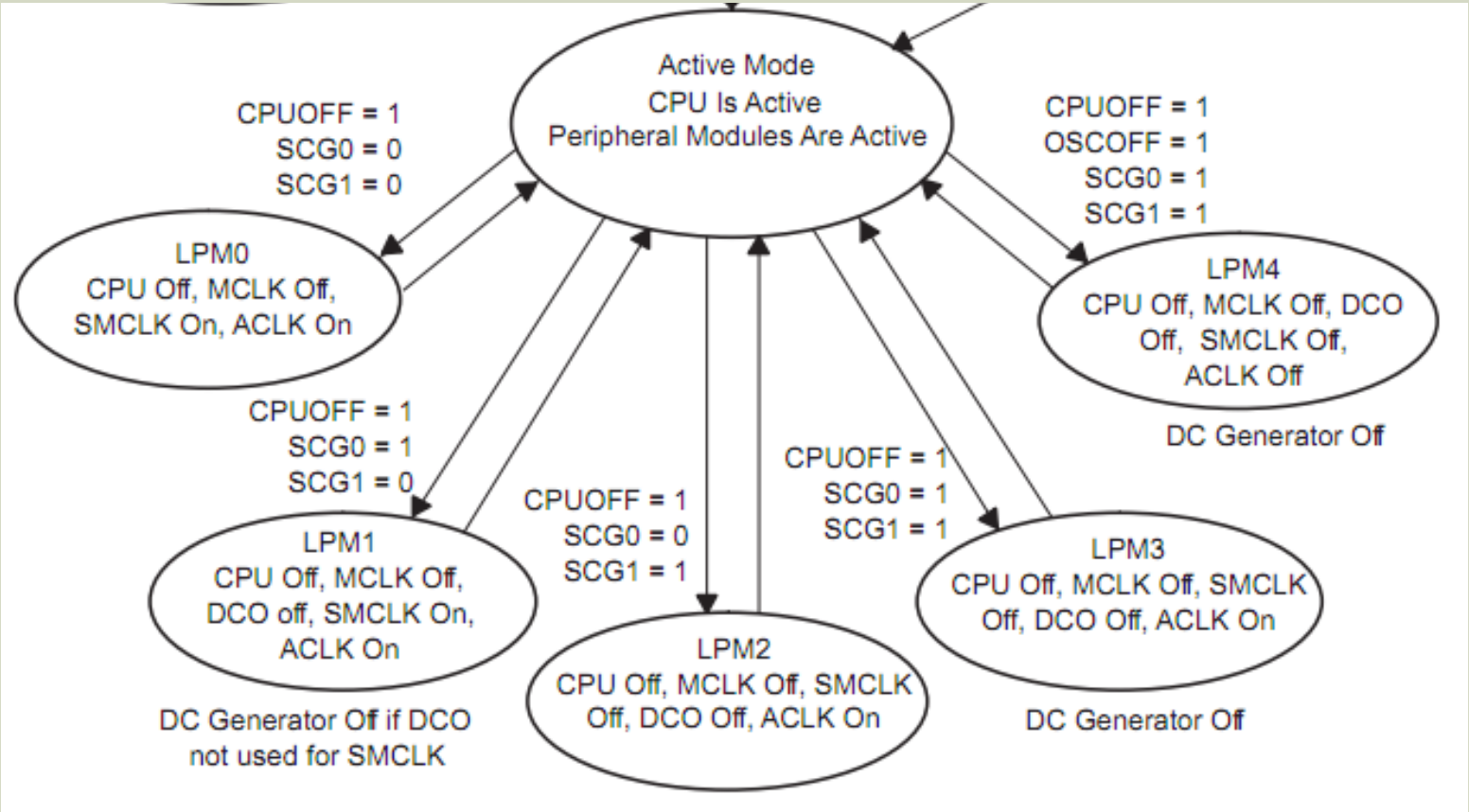
- In today's battery driven world, power saving is a must for many consumer based applications
- Most modern day microprocessors/controllers have a variety of onboard power-saving features
  - Compare TI vs. Microchip
- TI
  - MSP430 Line of "Ultra Low Power MCU's"
- Microchip
  - PIC nanoWatt eXtreme Low Power MCU's

VS

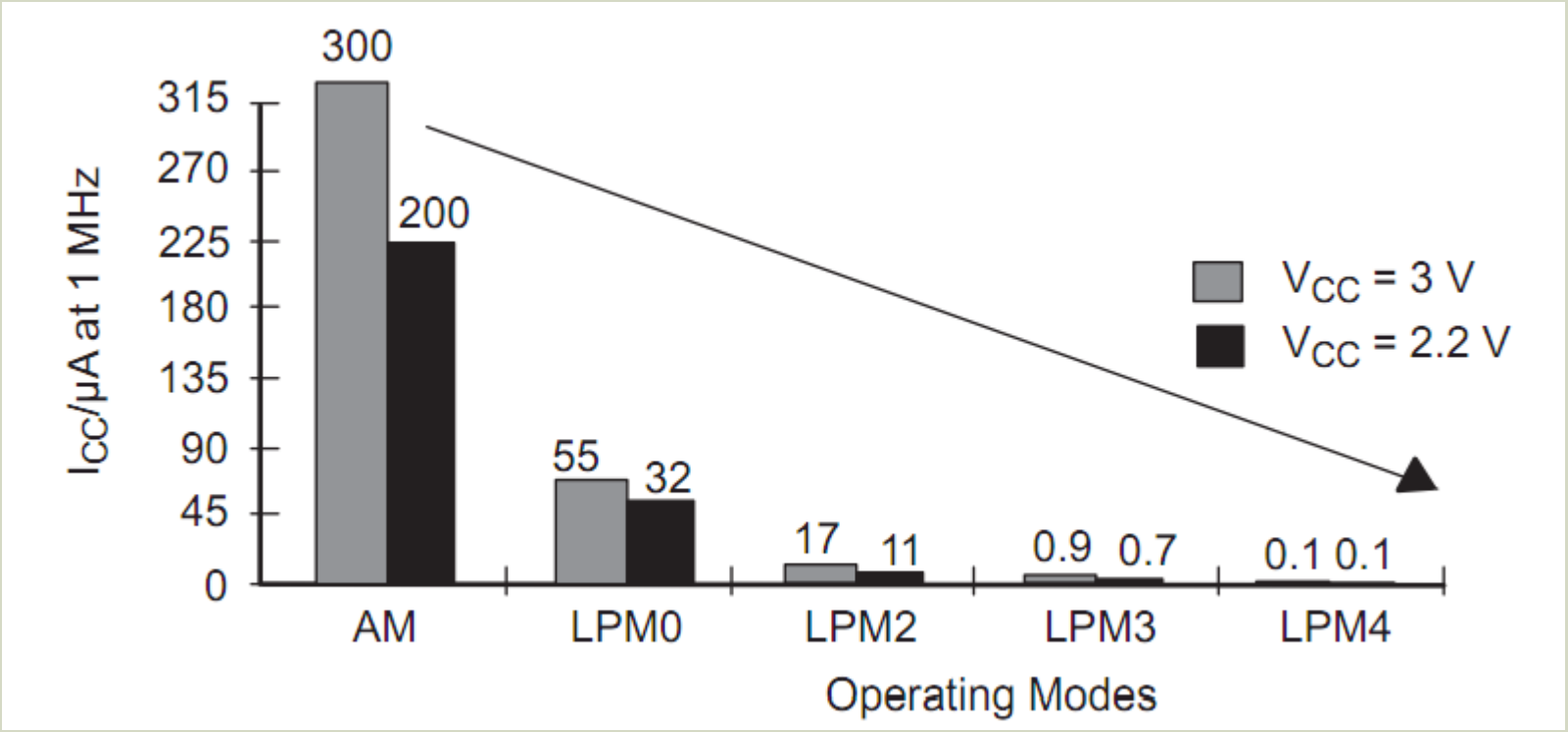
MSP430 – LOW POWER MODES (LPMs)

- Active
- LPM0 – CPU, Master Clock are disabled
- LPM1 – Same as LPM0, with the DCO and DC generator disabled if they aren't used for the Sub-System Master Clock
- LPM2 – Only DC Generator and the Auxiliary Clock are active
- LPM3 – Only Auxiliary Clock is active
- LPM4 – All clocks disabled; CPU disabled.

MSP430 – OPERATING MODES FLOWCHART



MSP430 – OPERATING MODES



PIC nanoWatt– OPERATING MODES

TABLE 1: POWER-SAVING OPERATING MODES FOR nanoWatt TECHNOLOGY DEVICES

Operating Mode	Active Clocks	Active Peripherals	Wake-up Sources	Typical Current	Typical Usage
Deep Sleep(1)	• Timer1/SOSC • INTRC/LPRC	• RTCC • DSWDT • DSBOR • INT0	• RTCC • DSWDT • DSBOR • INT0 • $\overline{MCLR}$	< 50 nA	• Long life, battery-based applications • Applications with increased Sleep times(3)
Sleep	• Timer1/SOSC • INTRC/LPRC • A/D RC	• RTCC • WDT • ADC • Comparators • CVREF • INTx • Timer1 • HLVD • BOR	All device wake-up sources (see device data sheet)	50-100 nA	Most low-power applications
Idle	• Timer1/SOSC • INTRC/LPRC • A/D RC	All Peripherals	All device wake-up sources (see device data sheet)	25% of Run Current	Any time the device is waiting for an event to occur (e.g., external or peripheral interrupts)
Doze(2)	All Clocks	All Peripherals	Software or interrupt wake-up	35-75% of Run Current	Applications with high-speed peripherals, but requiring low CPU use
Run	All Clocks	All Peripherals	N/A	See device data sheet	Normal operation

Note 1: Available on PIC18 and PIC24 devices with nanoWatt XLP™ Technology only.

2: Available on PIC24, dsPIC and PIC32 devices only.

3: Refer to **"Deciding Between Sleep and Deep Sleep"** for guidance on when to use Sleep or Deep Sleep modes.

PIC VS. MSP430

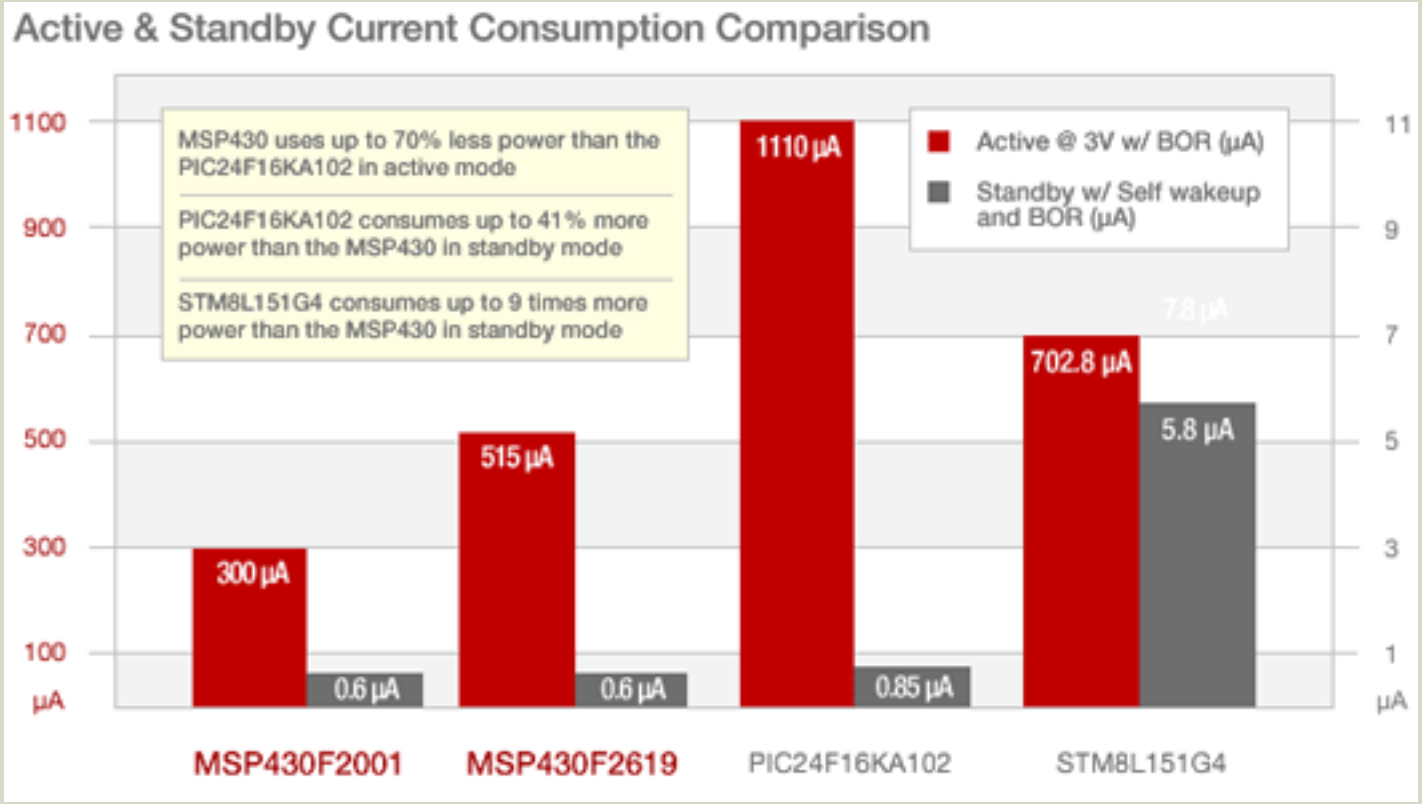
	Sleep Mode	BOR	WDT	RAM	Timer/ RTC	I/O State Maintained	Wake-up Time
MSP430	LPM3	Yes	Yes	Yes	Yes	Yes	Fast
	LPM4	Yes	No	Yes	No	Yes	Fast
	LPM5	No	No	No	No	No	Extended
PIC MCU w/XLP	Sleep	Yes	Yes	Yes	Yes	Yes	Fast
	Deep Sleep	Yes	Yes	No	Yes	Yes	Extended

APPLICATION NOTE EXAMPLES

- Both TI, and Microchip tested their processors against each other
- TI found that the MSP430 line was most efficient, while Microchip found their nanoWatt line to be the most efficient
  - Application should be deciding factor

MSP430

- Stable, high speed clock
- Interrupts vs. polling
- Design for low



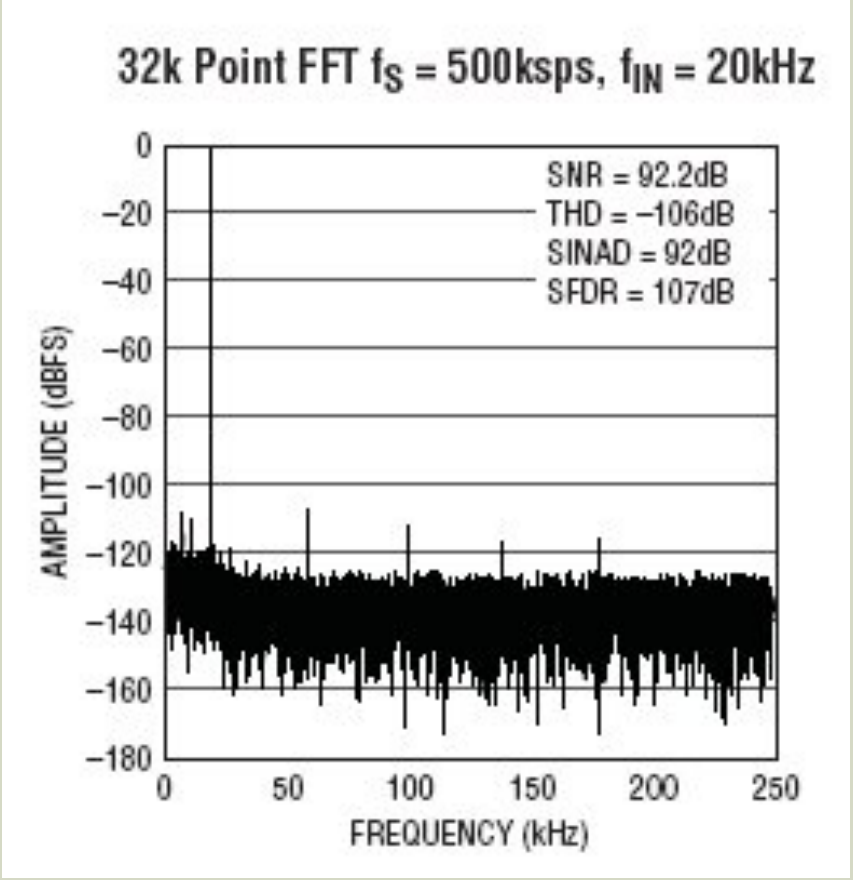
MSP430

- 16-bit Architecture is actually more efficient
- @ 1 MHz, MSP430 needs only 6us vs. 24us for the 8-bit competition
- Less code → less power

16-Bit MCU	8-Bit MCU
mov.w &ADC10MEM,&RAM	movf ADRESH,W
	movwf RAML
	bsf 0x20
	movlf ADCHRESL,W
	bcf 0x20
	movwf RAMH

LTC2382-16

- Low Noise, Low Power ADC: 6.5mW at 500ksps
- Internal Timer
- Low Power ADC Drivers
- Power down when not converting



QUESTIONS