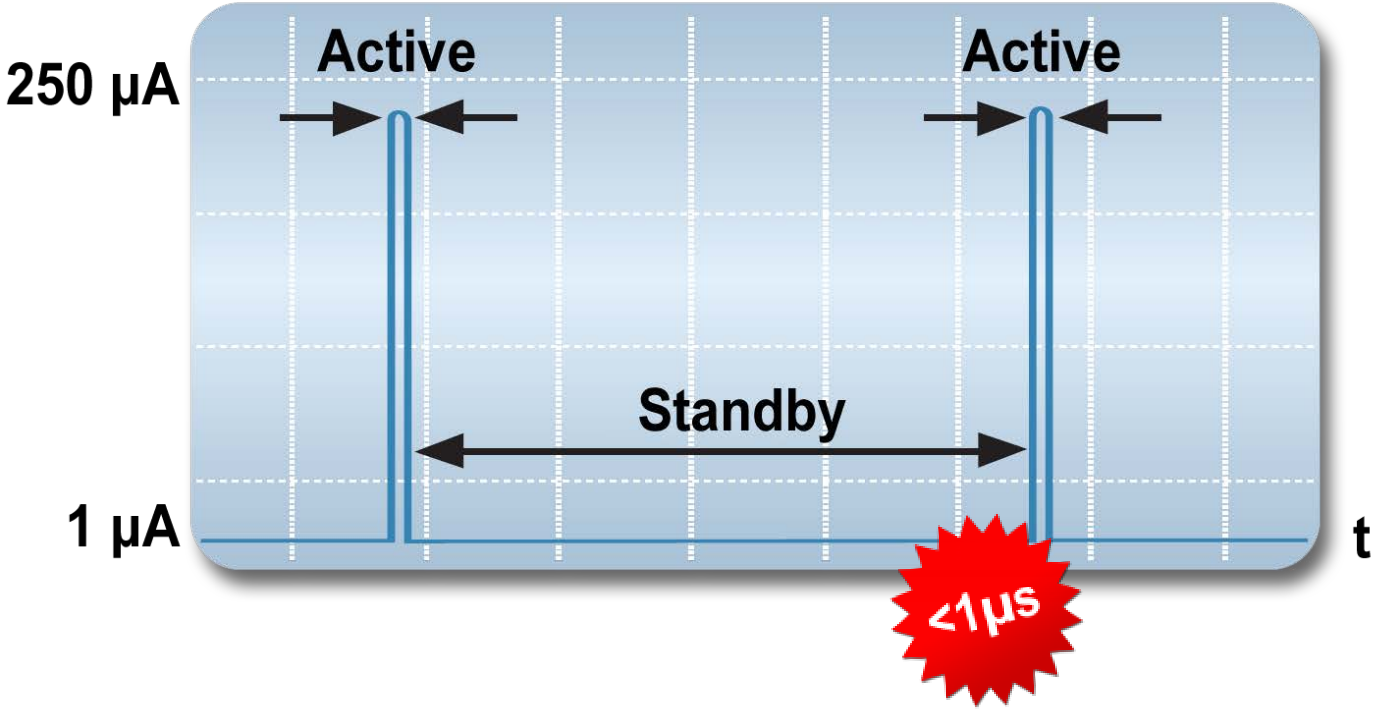
EEDG/CE 6302

Lecture 5:

Clocks and Low Power Modes

Some slides and photos courtesy of Texas Instruments

Ultra-low Power Activity Profile



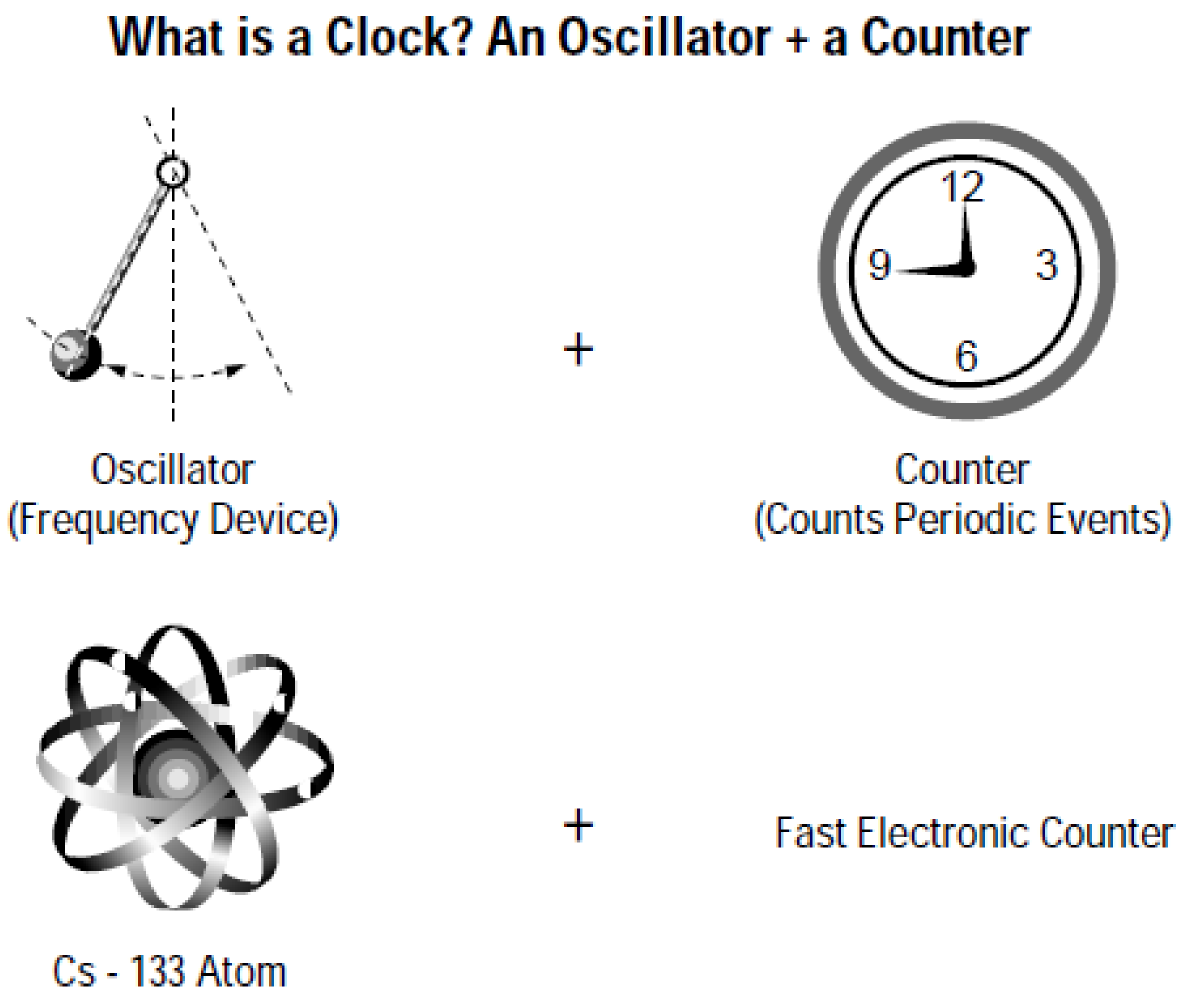
- Extended *Ultra-low Power* standby mode
- Minimum active duty cycle
- Interrupt driven performance on-demand

Why *Ultra-low Power* Is Important

- Longer battery life
- Smaller products
- Simpler power supplies
- Less EMI simplifies PCB
- *Permanent* battery
- Reduced liability

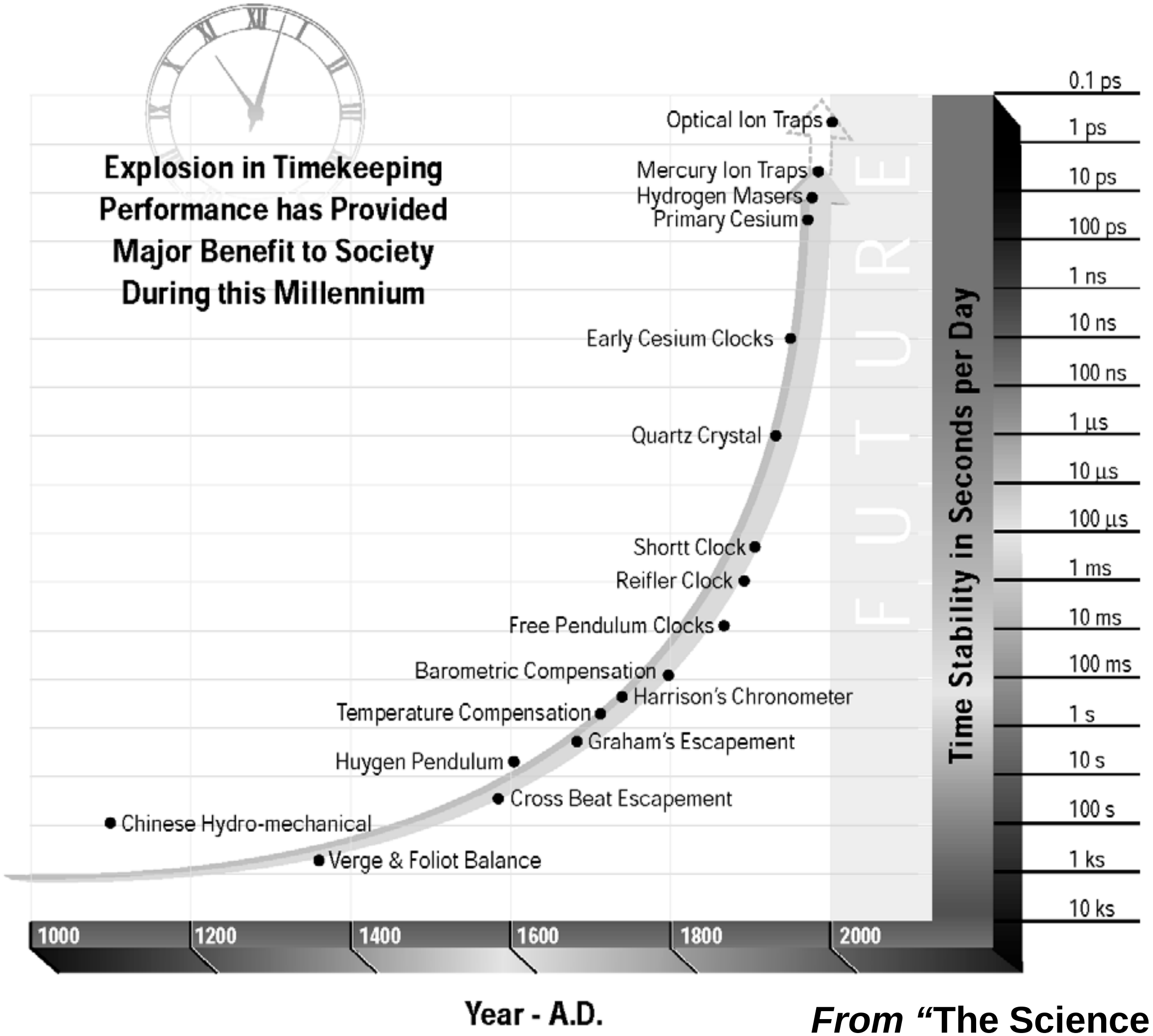
What is a Clock?

Clock



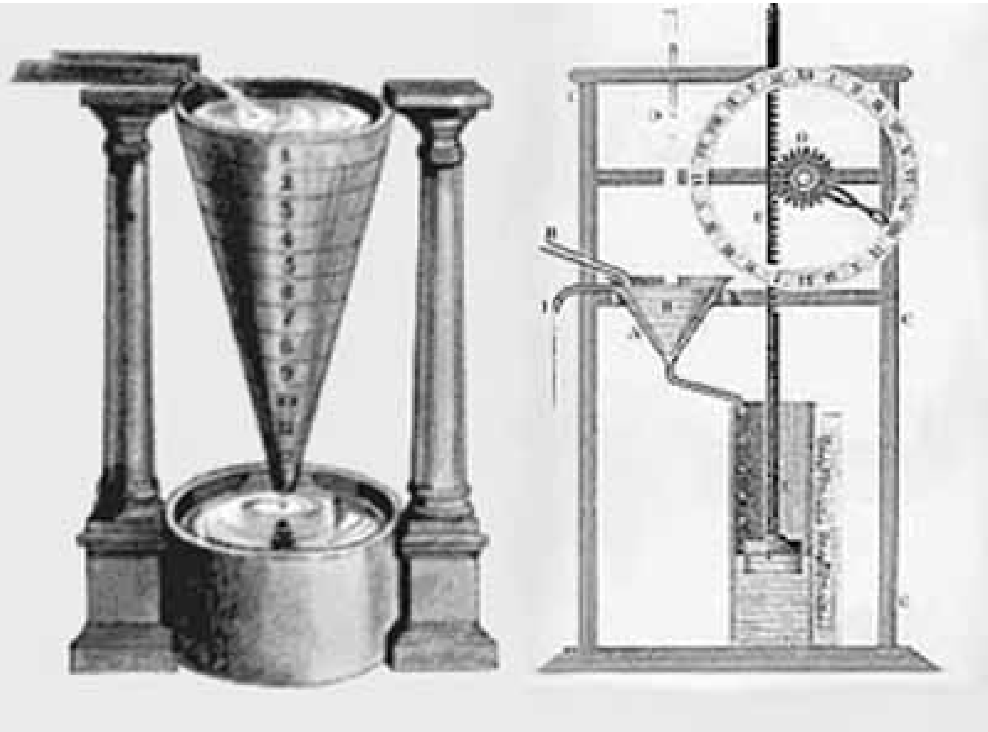
***From* “The Science of Timekeeping”**

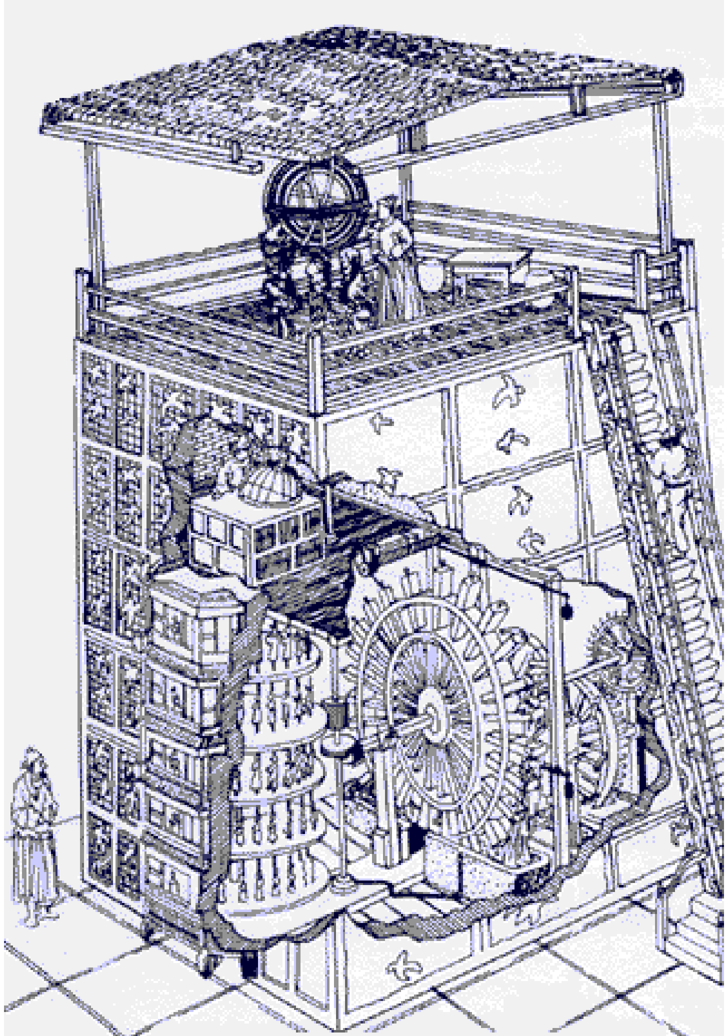
Time Keeping



From **"The Science of Timekeeping"**

Water Clocks





Su Sung water clock tower

Clocks and Low Power Modes

MSP430

MSP430 Basic Clock Module

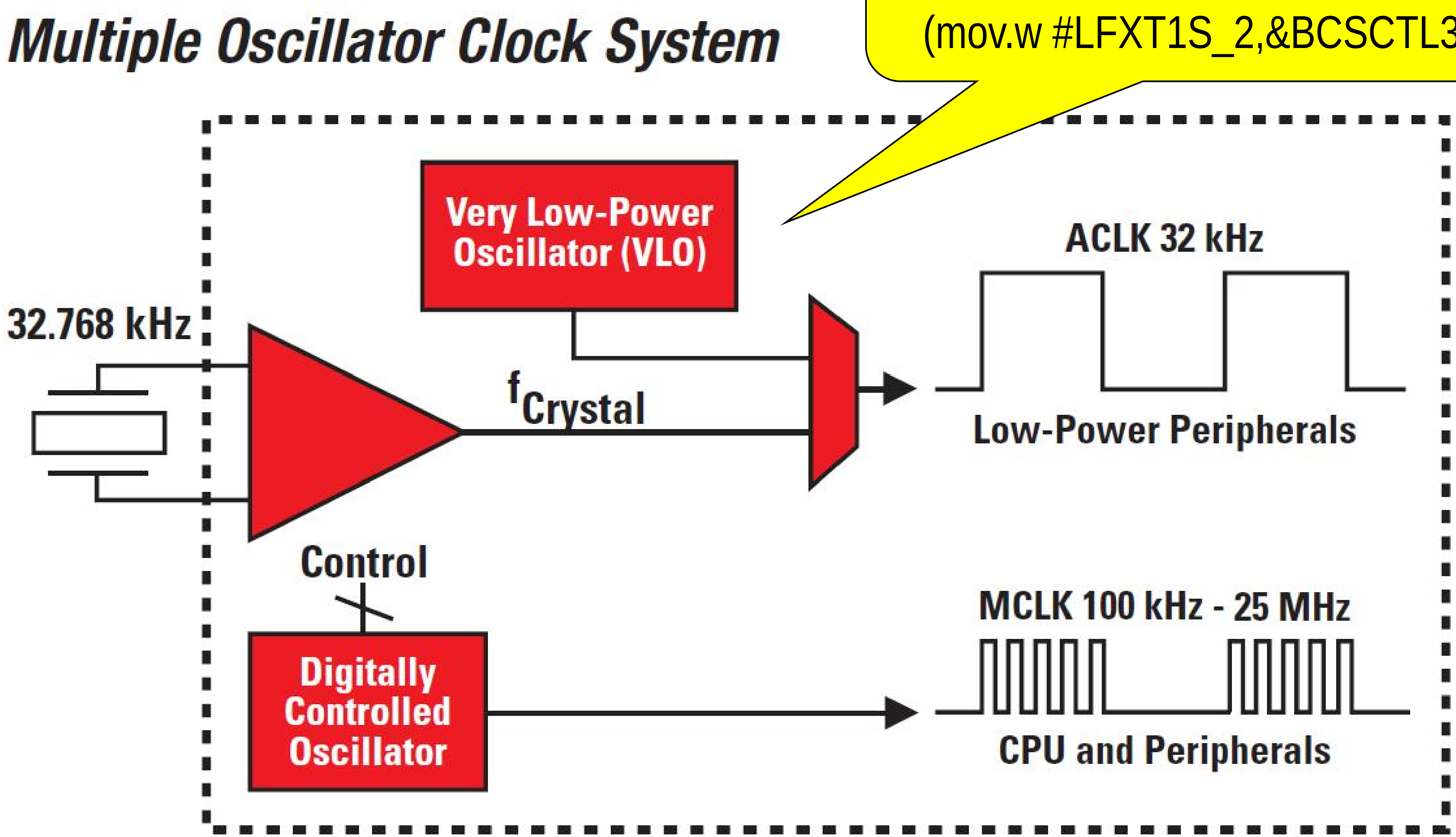
- Clock Sources:
  - LFXT1CLK: Low-frequency/high-frequency oscillator
  - T2CLK: Optional high-frequency oscillator
  - DCOCLK: Internal digitally controlled oscillator (DCO)

MSP430 Basic Clock Module

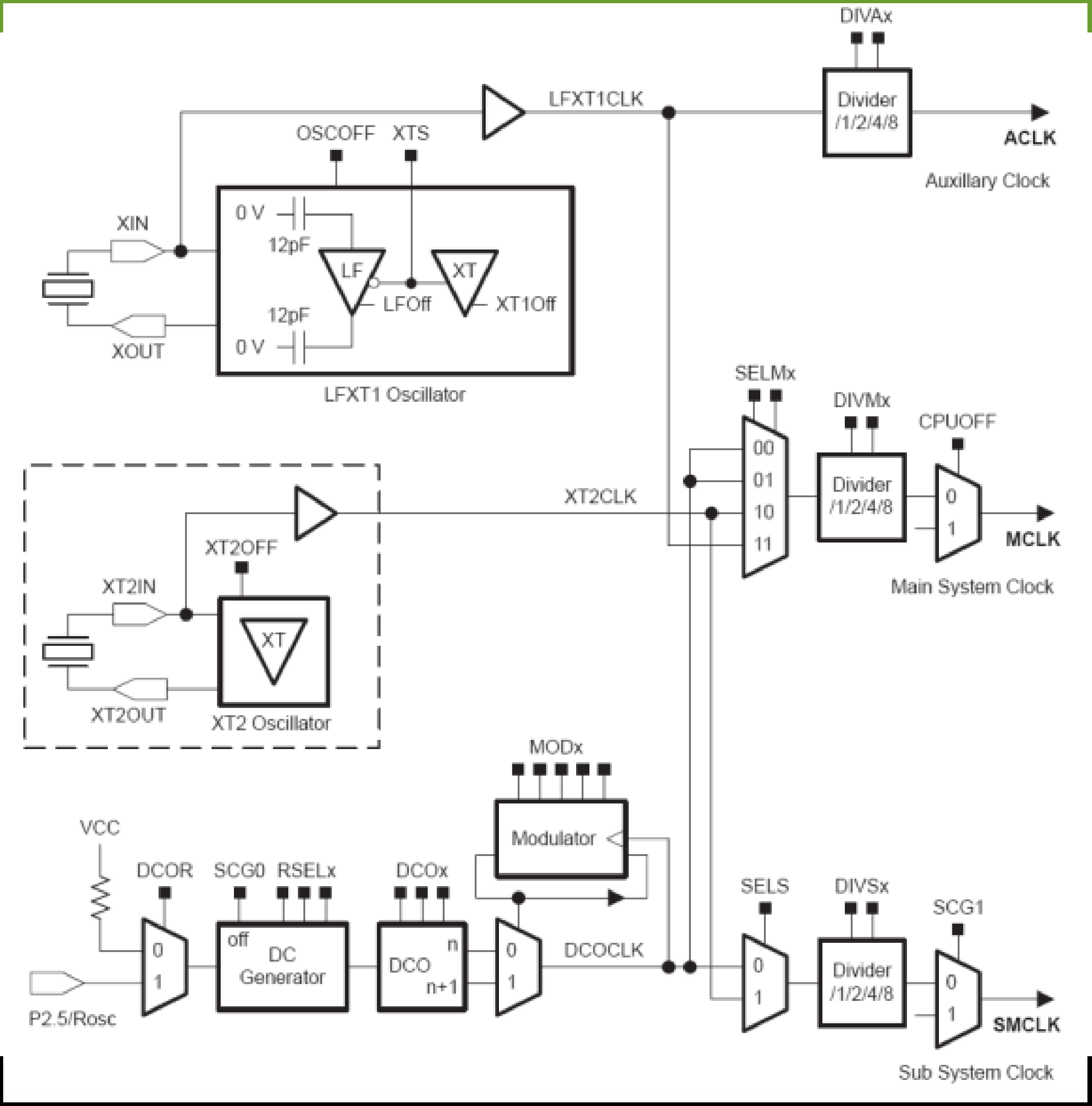
- Clock Signals:
  - ACLK: Auxiliary clock. The signal is sourced from LFXT1CLK with a divider of 1, 2, 4, or 8. (The calibration program for the serial link sets the divider to 4, but after the calibration it can be changed to any other values.) ACLK can be used as the clock signal for Timer A and Timer B.
  - MCLK: Master clock. The signal can be sourced from LFXT1CLK, XT2CLK (if available), or DCOCLK with a divider of 1, 2, 4, or 8. MCLK is used by the CPU and system.
  - SMCLK: Sub-main clock. The signal is sourced from either XT2CLK (if available), or DCOCLK with a divider of 1, 2, 4, or 8. SMCLK can be used as the clock signal for Timer A and Timer B.

Multiple Clocks

No crystal on eZ430 tools
Use VLO for ACLK
(mov.w #LFXT1S_2,&BCSCTL3)



MSP430 Clocking Circuit



Processor Clock Speeds

- Often, the most important factor for reducing power consumption is slowing the clock down
  - Faster clock = Higher performance, more power
  - Slower clock = Lower performance, less power
- Using assembly code:

```
;   MSP430 Clock - Set DCO to 8 MHz:
    mov.b   #CALBC1_8MHZ,&BCSCTL1    ; Set range
    mov.b   #CALDCO_8MHZ,&DCOCTL     ; Set DCO step + modulation
```

- Using C code:

```
//   MSP430 Clock - Set DCO to 8 MHz:
    BCSCTL1 = CALBC1_8MHZ;    // Set range 8MHz
    DCOCTL = CALDCO_8MHZ;     // Set DCO step + modulation
```

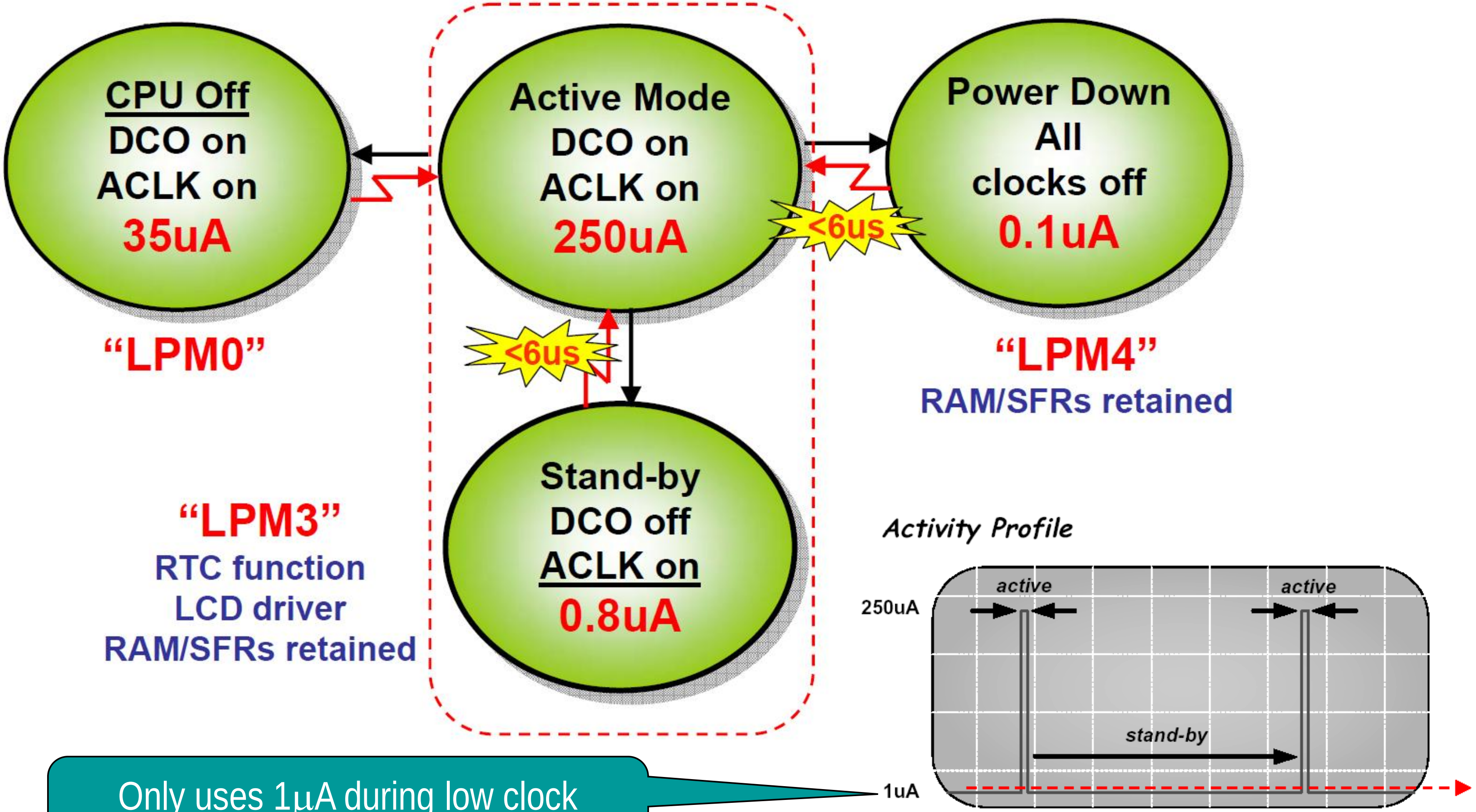
Low Power Modes

- The TI MSP430 has four power mode/levels.
- Each has different characteristics for:
  - Memory state
  - Instruction Execution state
  - IO state
  - Timer/Clock state
- The MSP430 is highly optimized for low power, further enabled by the proper usage of the low power modes of the device.

Processor Clock Speeds

- Another method to reduce power consumption is to turn off some (or all) of the system clocks
  - **Active Mode (AM)**: CPU, all clocks, and enabled modules are active (≈300 μA)
  - **LPM0**: CPU and MCLK are disabled, SMCLK and ACLK remain active (≈85 μA)
  - **LPM3**: CPU, MCLK, SMCLK, and DCO are disabled; only ACLK remains active (≈1 μA)
  - **LPM4**: CPU and all clocks disabled, RAM is retained (≈0.1 μA)
- A device is said to be ***sleeping*** when in low-power mode; ***waking*** refers to returning to active mode

MSP430 Clock Modes



Only uses 1µA during low clock
Less clocks means less power!

MSP430 Clock Settings

Reserved	V	SCG1	SCG0	OSC OFF	CPU OFF	GIE	N	Z	C

R2/SR

	SCG1	SCG0	OSC OFF	CPU OFF	
Active Mode	0	0	0	0	~ 250uA
LPM0	0	0	0	1	~ 35uA
LPM3	1	1	0	1	~ 0.8uA
LPM4	1	1	1	1	~ 0.1uA

SMCLK and ACLK Active

Only ACLK Active

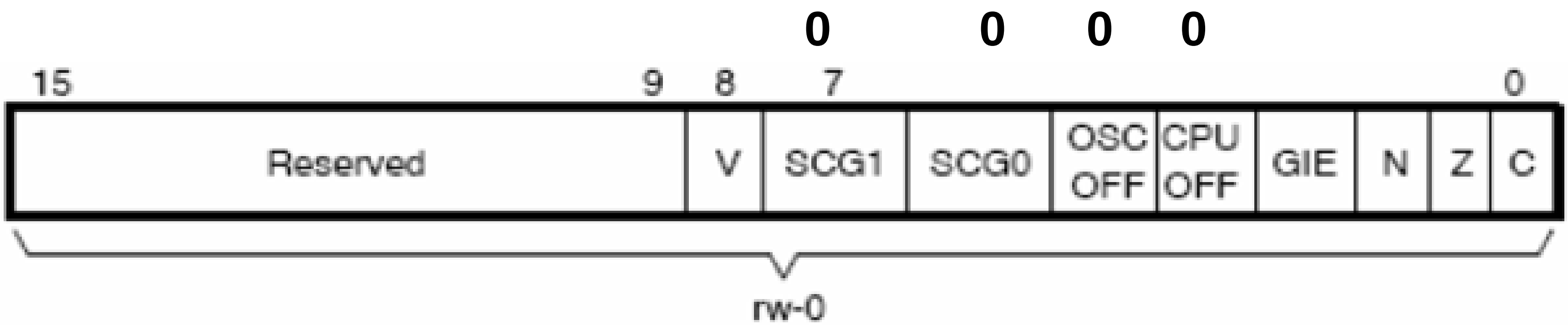
No Clocks!

Sleep Modes

```
bis.w    #CPUOFF,SR             ; LPM0
```

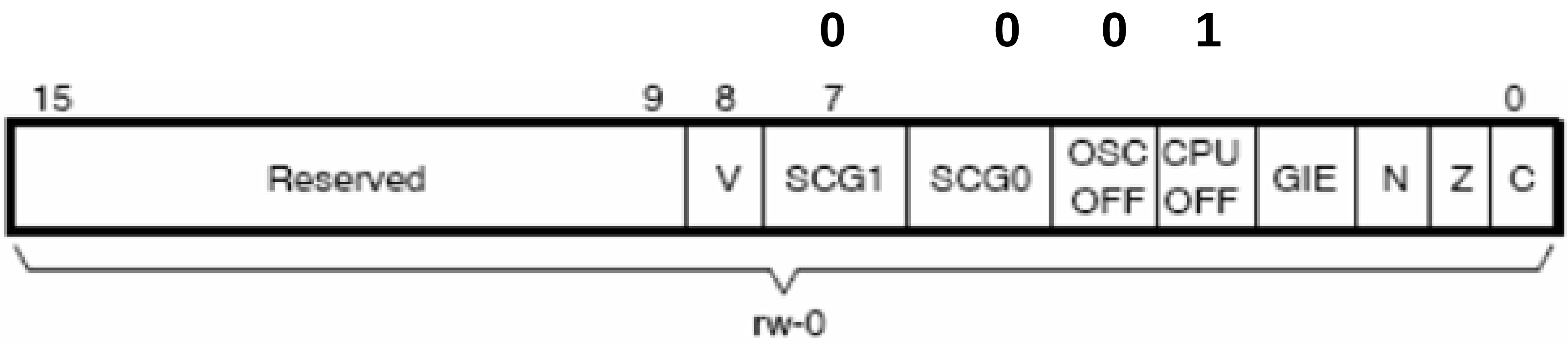
Low Power Modes - Active Mode

- All clocks of the MSP430 are active
- MSP430 starts up in this mode, and enters this mode when an interrupt is detected
- Only mode where instructions are executing.
- Current can range from 200 to 300 µA



Low Power Modes - LPM0

- LPM0 has the CPU and MCLK disabled.
- Used in cases where active peripherals need a high speed SMCLK.
- Current is approximately 85 µA



Low Power Modes - LPM3

- CPU, MCLK, SMCLK, and DCO are disabled
- ACLK remains active.
- This is the standard low power mode when the MSP430 must awaken itself at regular time intervals, and therefore needs a slow clock.
- If the RTC is to be maintained this is the necessary low power mode.
- Current is approximately 1 µA

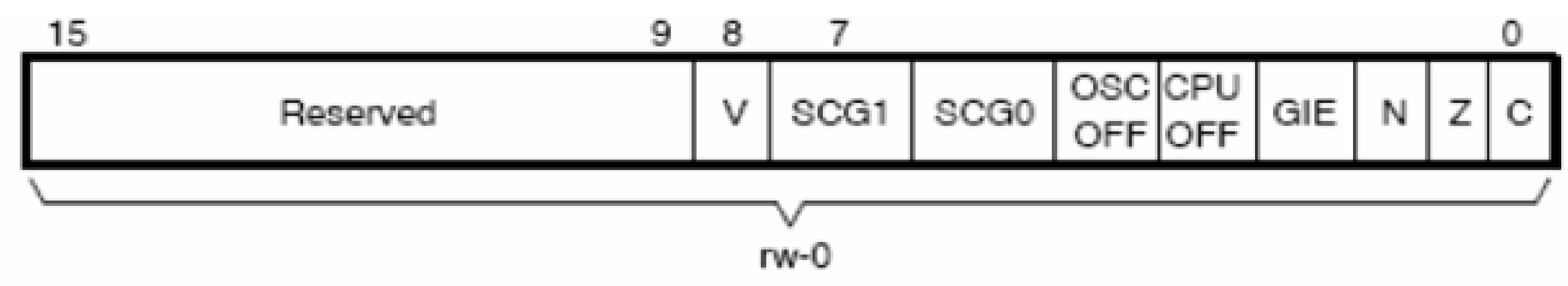
		1	1	0	1				
15 … 9	8	7							0
Reserved	V	SCG1	SCG0	OSC OFF	CPU OFF	GIE	N	Z	C

rw-0

Low Power Modes - LPM4

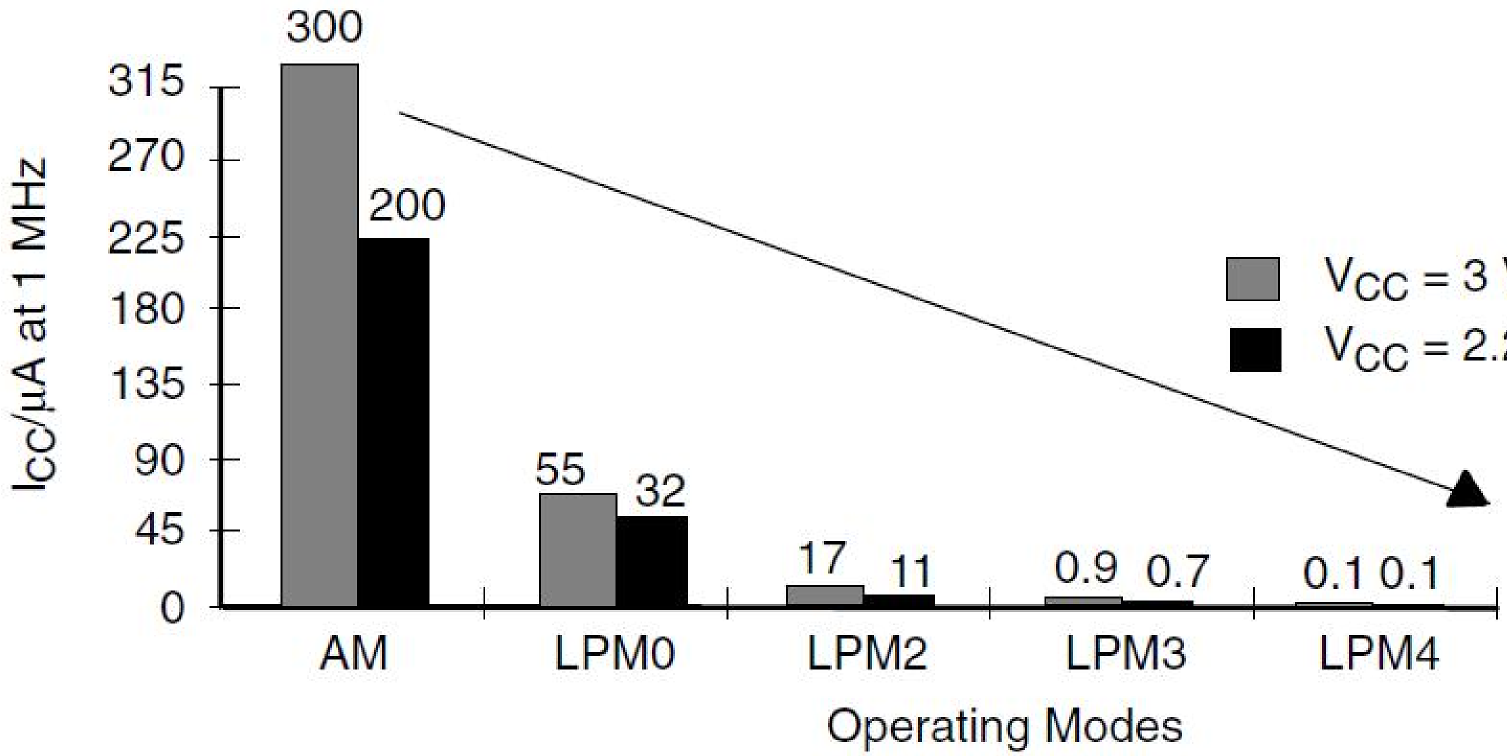
- CPU and all clocks are disabled
- The MSP430 can only awaken from and external signal.
- The mode is know as *RAM retention mode*
- Current is approximately 0.1 µA

1 1 1 1



Lower Power Savings

- Finally, powering your system with lower voltages means lower power consumption as well



Setting Low-Power Modes

- Setting low-power mode puts the microcontroller "to sleep" – so usually, interrupts would need to be enabled as well.
- Enter LPM3 and enable interrupts using assembly code

```
;   enable interrupts / enter low-power mode 3
    bis.b   #LPM3+GIE,SR      ; LPM3 w/interrupts
```

- Enter LPM3 and enable interrupts using C code:

```
//   enable interrupts / enter low-power mode 3
    __bis_SR_register(LPM3_bits+GIE);
```

Principles of Low-Power Apps

- Maximize the time in LPM3 mode
- Use interrupts to wake the processor
- Switch on peripherals only when needed
- Use low-power integrated peripherals
  - Timer_A and Timer_B for PWM
- Fast table look-ups instead of calculations
- Avoid frequent subroutine and function calls
- Longer software routines should use single-cycle CPU registers

Waking from an LPM (Interrupt Handling)

- An interrupt is necessary to exit a LPM.
- The MCLK and CPU are started automatically to allow the CPU to service the interrupt.
- Since the SR is cleared when an interrupt occurs it is necessary to modify the SR if you wish to return to normal operation after the interrupt service routine exits.
- If all processing necessary for service is completed in the interrupt the CPU will return to the low power mode upon exit.

Returning to the Main Function from a LPM

- To return the Main function after an interrupt wakes the MSP430 from a LPM you must:
  - Clear all the LPM bits in the *SAVED value of SR on the stack*.
  - In C an intrinsic function __low_power_mode_off_on_exit() handles this for you.
  - In Assembly the programmer must take this burden directly. (The SP will be pointing at the SR on the stack upon entry to the ISR)
- Upon exit the modified SR will be restored and normal operation in Active mode will resume.

Class Exercise

Write a C program that initializes an interrupt on change of input pin P1.4. When the interrupt occurs, the input on P2.0-1 will be inverted and output on P1.0-1. Count the number of transitions.

When waiting for the interrupt to occur the MSP430 should be in the appropriate LPM that will allow the output to be maintained with the least power consumption.

Previous Example – Part I

```
#include "io430.h"
#include "intrinsics.h"

integer count;                          // (Global Variable)

void main( void )
{
  WDTCTL = WDTPW | WDTHOLD;             // Stop watchdog timer to prevent time out reset
  P1DIR = BIT0 | BIT1;                  // set p1.0-1 to output, P1.4 to input
  P2DIR = 0x00;                         // set all pins on port P2 to input
  P1IE = BIT4;                          // enable interrupts for p1.4 (default rising edge)
  __enable_interrupt();                 // enable global interrupts

  count = 0;

for (;;)
  {
    __low_power_mode_?();               // go to low power mode
  }
}
```

Previous Example – Part II

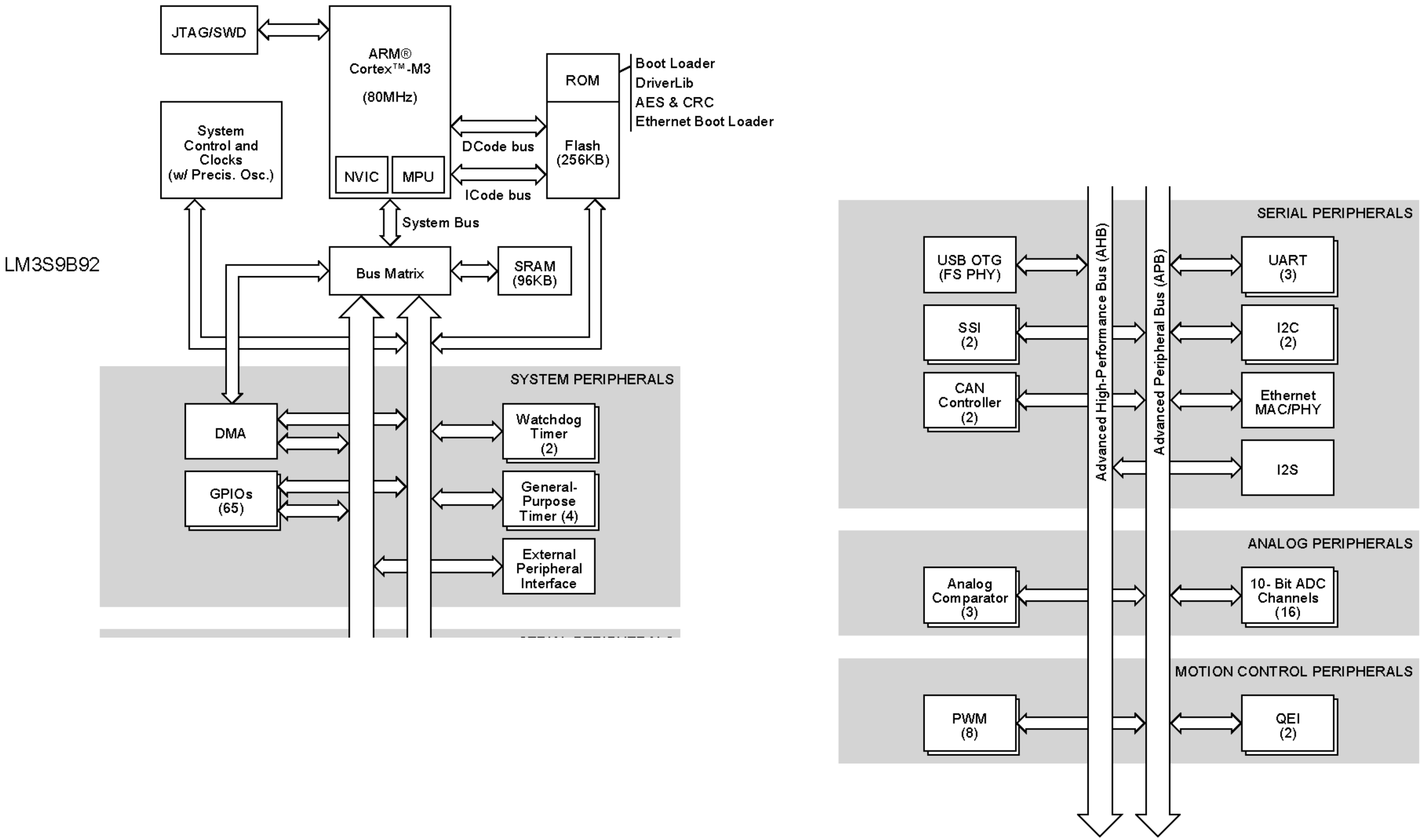
```
// In the same file as Part I

#pragma vector = PORT1_VECTOR
__interrupt void p1_isr()
{
  if (P1IN & BIT4)  {
     P1IFG &= ~BIT4;                        // clear the interrupt flag
     P1OUT = ~ ((P2IN & BIT0) | (P2IN & BIT1))
                                            // invert P2.0-1 and send to P1.0-1
     count++;
}
```

Clocks and Low Power Modes

ARM Cortex-M3

LM3S9B92 Block Diagram (Evalbot)



Clock Sources

- Precision Internal Oscillator (PIOSC)
  - The precision internal oscillator is an on-chip clock source that is the clock source the microcontroller uses during and following POR. It does not require the use of any external components and provides a clock that is 16 MHz ±1% at room temperature and ±3% across temperature. The PIOSC allows for a reduced system cost in applications that require an accurate clock source. If the main oscillator is required, software must enable the main oscillator following reset and allow the main oscillator to stabilize before changing the clock reference.

Clock Sources

- Main Oscillator (MOSC)
  - The main oscillator provides a frequency-accurate clock source by one of two means: an external single-ended clock source is connected to the OSC0 input pin, or an external crystal is connected across the OSC0 input and OSC1 output pins. If the PLL is being used, the crystal value must be one of the supported frequencies between 3.579545 MHz to 16.384 MHz (inclusive). If the PLL is not being used, the crystal may be any one of the supported frequencies between 1 MHz to 16.384 MHz. The single-ended clock source range is from DC through the specified speed of the microcontroller.

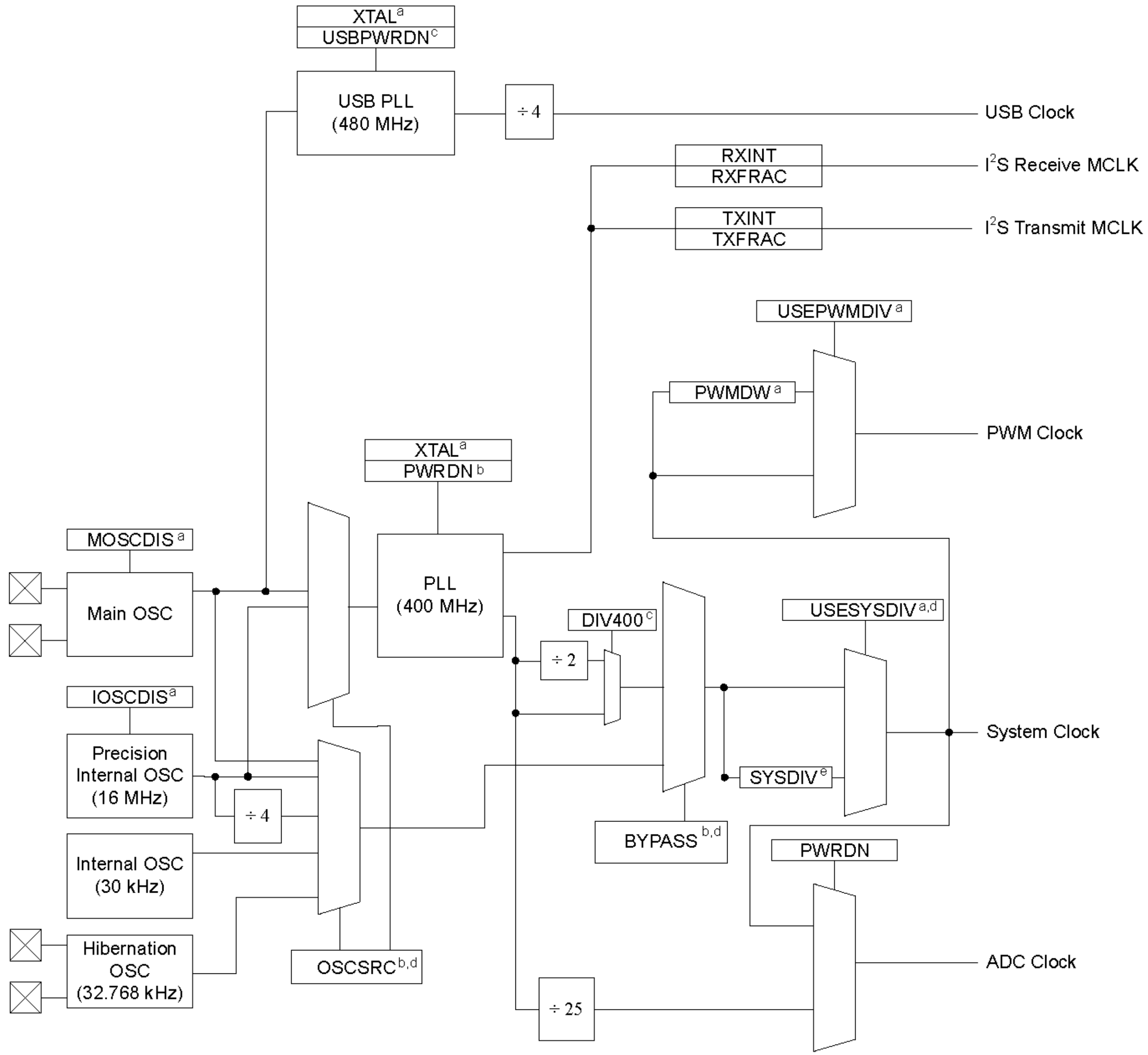
Clock Sources

- Internal 30-kHz Oscillator
  - The internal 30-kHz oscillator provides an operational frequency of 30 kHz. It is intended for use during Deep-Sleep power-saving modes. This power-savings mode benefits from reduced internal switching and also allows the MOSC to be powered down.

Hibernation OSC

- Hibernation OSC 32.768 KHz
  - This battery-backed oscillator is used for the hibernation module that includes a real-time clock, a 256 bytes of non-volatile battery-backed memory, low-battery detection, signaling, and interrupt detection, and a hibernate mode that can wake on a real-time clock match, external pin interrupt, or low-battery event. With a hibernate mode including real-time clock consuming approximately 16 uA standby current, a standard CR2032 watch battery can support a Stellaris system in hibernate mode for over 3 years.

Clock Tree


XTAL[a]
USBPWRDN[c]
USB PLL (480 MHz)
÷ 4
USB Clock
RXINT
RXFRAC
I²S Receive MCLK
TXINT
TXFRAC
I²S Transmit MCLK
USEPWMDIV[a]
PWMDW[a]
PWM Clock
XTAL[a]
PWRDN[b]
MOSCDIS[a]
Main OSC
PLL (400 MHz)
DIV400[c]
USESYSDIV[a,d]
÷ 2
IOSCDIS[a]
Precision Internal OSC (16 MHz)
÷ 4
SYSDIV[e]
System Clock
BYPASS[b,d]
PWRDN
Internal OSC (30 kHz)
Hibernation OSC (32.768 kHz)
OSCSRC[b,d]
÷ 25
ADC Clock

Modes of Operation

- Run mode
- Sleep mode
- Deep-Sleep mode
  - **RCGCn**, **SCGCn**, and **DCGCn** registers control the clock gating logic for each peripheral or block in the system while the microcontroller is in Run, Sleep, and Deep-Sleep mode, respectively.

Run Mode

- In Run mode, the microcontroller actively executes code. Run mode provides normal operation of the processor and all of the peripherals that are currently enabled by the **RCGCn** registers. The system clock can be any of the available clock sources including the PLL.

Sleep Mode

- In Sleep mode, the clock frequency of the active peripherals is unchanged, but the processor and the memory subsystem are not clocked and therefore no longer execute code. Sleep mode is entered by the Cortex-M3 core executing a WFI (Wait for Interrupt) instruction. Any properly configured interrupt event in the system brings the processor back into Run mode.

Deep Sleep Mode

- In Deep-Sleep mode, the clock frequency of the active peripherals may change (depending on the Run mode clock configuration) in addition to the processor clock being stopped. An interrupt returns the microcontroller to Run mode from one of the sleep modes; the sleep modes are entered on request from the code. Deep-Sleep mode is entered by first setting the SLEEPDEEP bit in the System Control (SYSCTRL).

Energy Harvesting

Available Energy is All Around

Light

Radio frequency

Motion and vibration

Heat

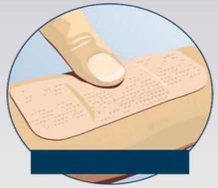
Energy Harvesting Basics

- ***Energy harvesting*** is the process by which energy is ***captured*** and ***stored***

- This term frequency refers to small autonomous devices – micro energy harvesting

- A variety of sources exist for harvesting energy
  - solar power
  - thermal energy
  - wind energy
  - salinity gradients
  - kinetic energy
  - radio frequency

No-Power Apps



Body warn monitoring devices powered by body heat, movement

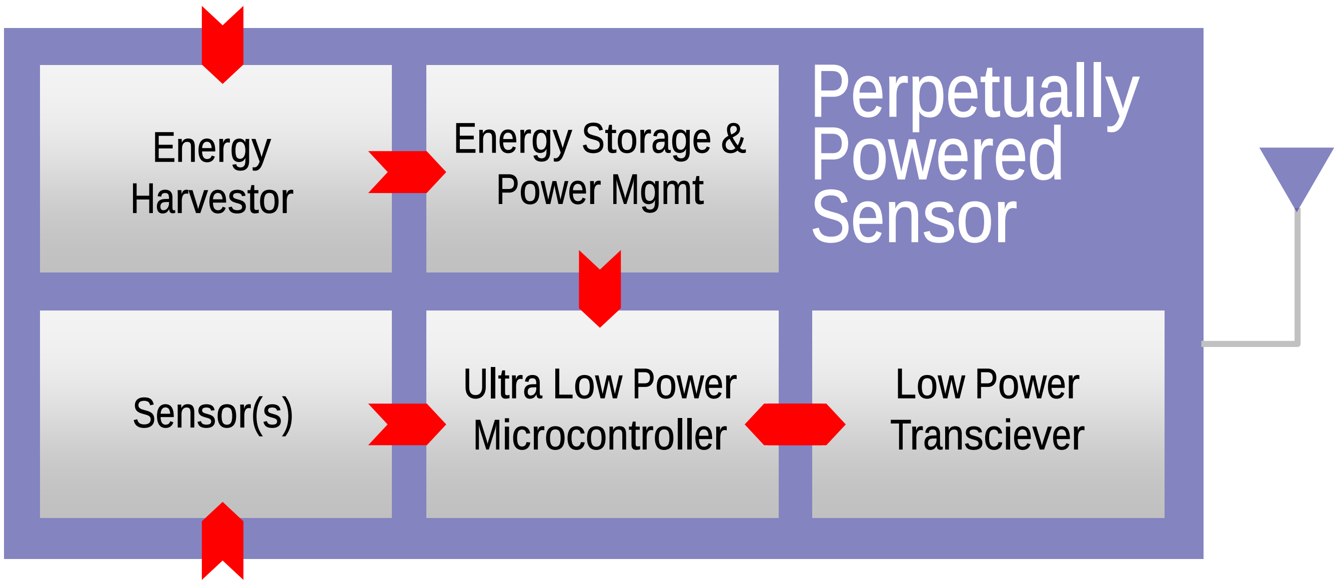
Monitor environmental conditions on farm, winery, etc.

Mesh networking for environmental monitoring (e.g. forest fire detection)

Automotive monitoring (e.g. tire pressure gauges powered by vibration)

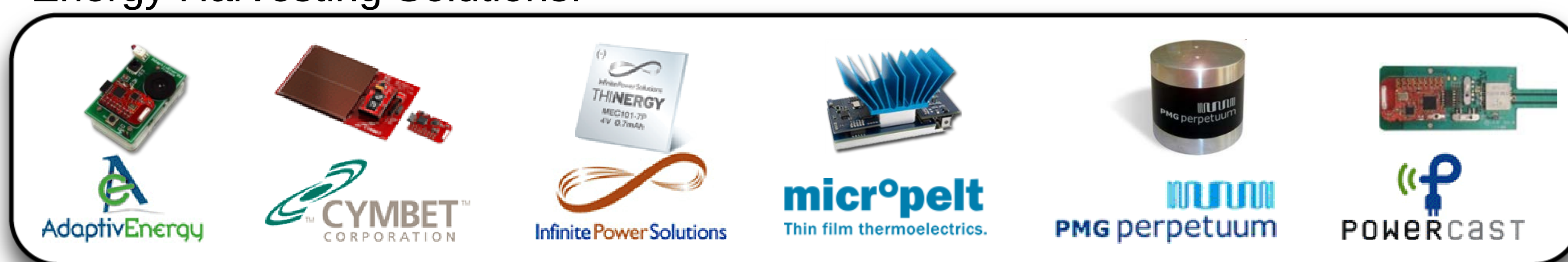
Energy Harvesting

Ambient energy: light, heat, motion, RF, etc

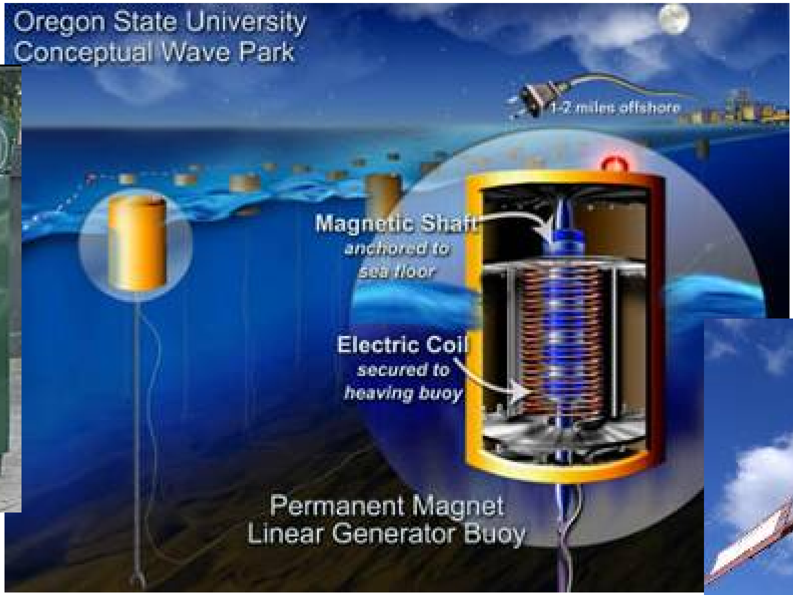


Environment: temperature, status, position, etc

Energy Harvesting Solutions:



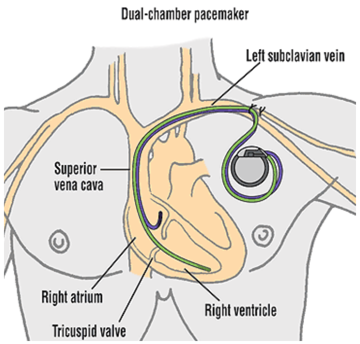
Energy Harvesting Isn't New



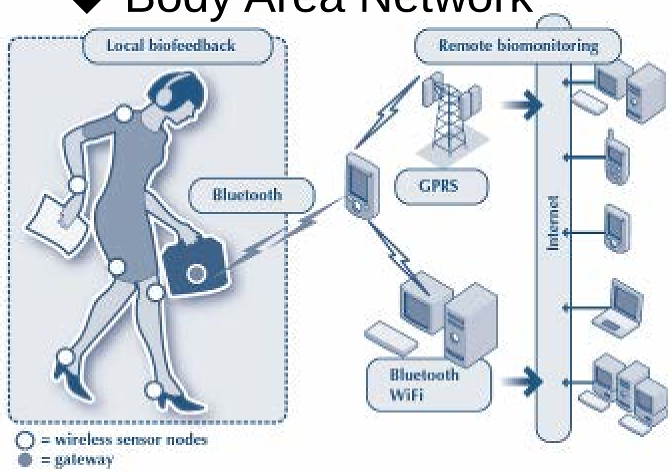
Energy Harvesting Applications

Low data rate, low duty cycle, ultra-low power

- Medical and Health monitoring



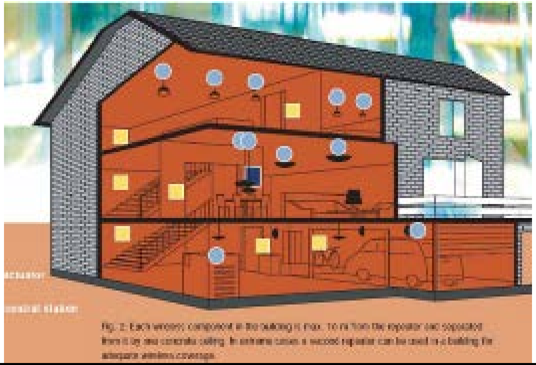
- Body Area Network



- Structure Health monitoring

- Wireless Sensor Networks

- Smart building



Energy Harvesting Tradeoffs

- **Advantages**
  - Mobile: no power wires
  - Easier installation
  - Lower maintenance
  - Environmentally friendly
  - Higher uptime
- **Disadvantages**
  - Dependent on availability of harvestable energy source
  - Strict power budget
  - Upfront cost may be higher
  - Less mature technology

When Does Harvesting Make Sense?

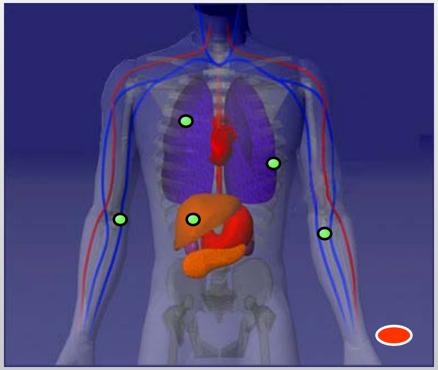
- Harvestable energy available
- Difficult to install or power devices
- Difficult to reach devices for maintenance
- Cords too costly
- Numerous devices
- Environmentally friendliness required
- High uptime demanded

One or more of these characteristics are required for energy harvesting to make sense compared to batteries

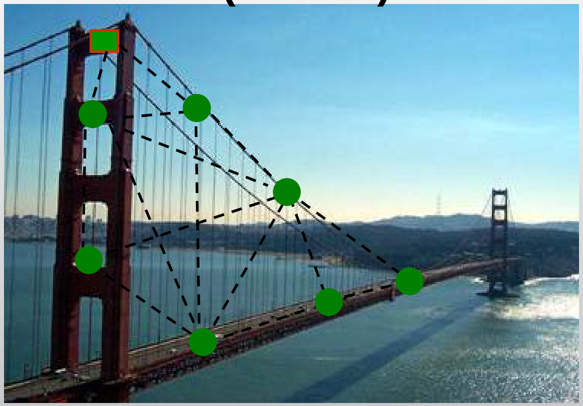
Permanently Powered Wireless Sensors

- Remote patient monitoring
- Harmful agents detection
- Efficient office energy control
- Surveillance and security
- Detecting and tracking enemy troop movement
- Vineyard or other agricultural management
- Home automation
- Implantable sensors
- Long range asset tracking
- Aircraft fatigue supervision

Remote patient monitoring (body heat)



Structural monitoring (motion)



Energy Harvesting Design Guides

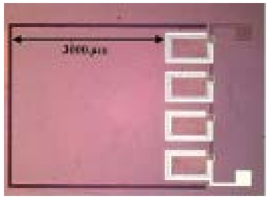
- Power budget – peak & standby
- Energy duty cycle
  - E_{in} vs. E_{out}
- Energy source
- Energy storage
- Operating condition
- Storage conditions
- Response time
- Cost of ownership

Energy Harvesting Sources

Energy Source	Characteristics	Efficiency	Harvested Power
Light	Outdoor Indoor	10~24%	100 mW/cm^2 100 µW/cm^2
Thermal	Human Industrial	~0.1% ~3%	60 µW/cm^2 ~1-10 mW/cm^2
Vibration	~Hz–human ~kHz–machines	25~50%	~4 µW/cm^3 ~800 µW/cm^3
RF	GSM 900 MHz WiFi	~50%	0.1 µW/cm^2 0.001 µW/cm^2

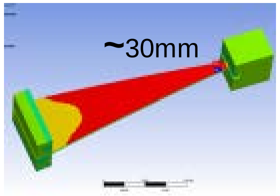
Seiko watch
~5uW

Holst Center
~40uW



2 channel EEG
~1mW

AdaptivEnergy
~10mW

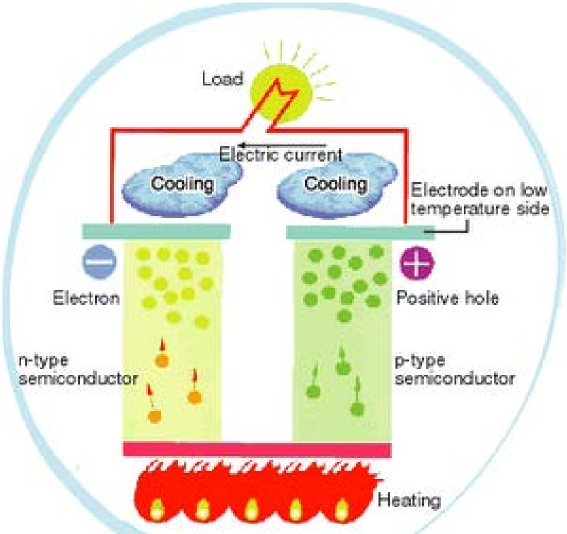


Elastometer
~800mW

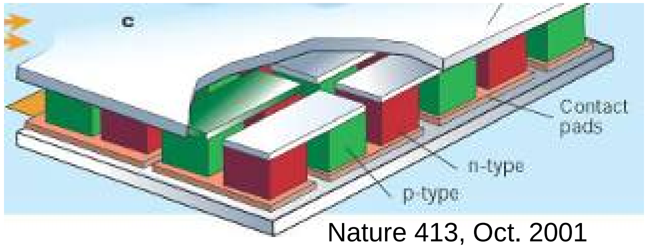
BigBelly
~40W

1uW **10uW** **100uW** **1mW** **10mW** **100mW** **1W+**

Harvesting Thermal Energy



Thermocouple



Nature 413, Oct. 2001

Thermopiles

- thermally in parallel
- electrically in serial

Thermoelectric Seebeck Effect

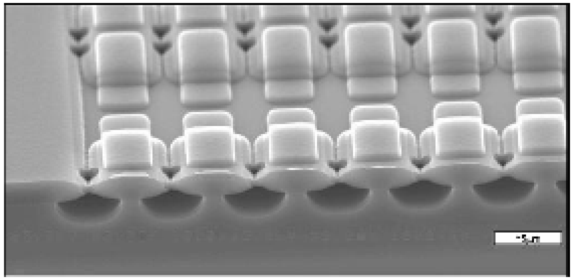
Temp. gradient drives heat flow

⇩

Electrons and holes flow in N-type and P-type lags made of semiconductor materials

⇩

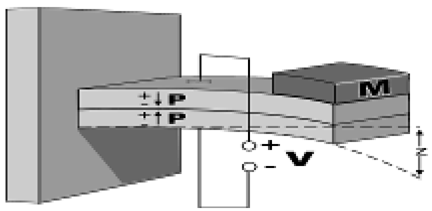
Carnot Efficiency
≡ ΔT/TH



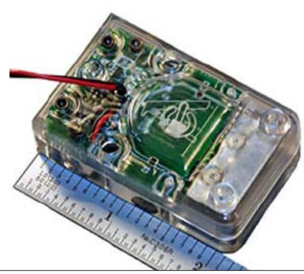
Si-thermocouples on chip

Harvesting Vibration Energy

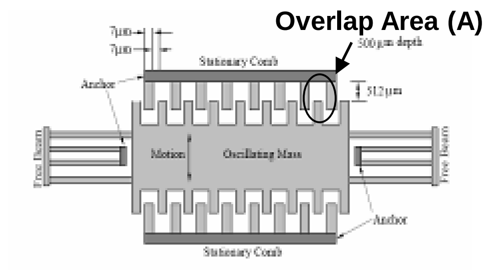
Piezoelectric



- Vibration → beam bending (strain)
- Piezoelectric material converts mechanical strain into electrical energy



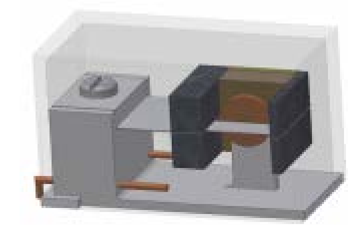
Electrostatic



- Vibration → motion of oscillating mass
- Comb overlap area (A) change
- Comb capacitance (C) change
- Voltage change at constant charge (Q)

$$C = \frac{\varepsilon_0 A}{d} \qquad Q = CV$$

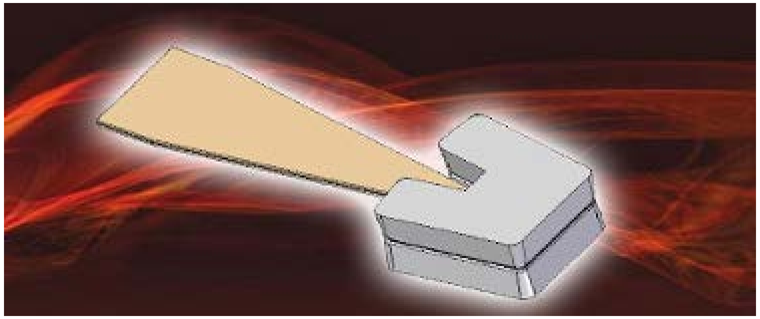
Electromagnetic



- Vibration → motion of magnetic field
- Current flows in the static copper coil

Vibration Solutions

- **AdaptivEnergy**
  - Highly efficient harvesting with periodic vibration
  - Higher energy output density with small form factor
  - Ability to customize to range of vibration frequencies



- **Perpetuum**
  - Vibration harvester
  - Sealed for rugged industrial environment application
  - Available today

Energy Harvesting Storage Required

- Scavenged energy is not constant
- Power not available on-demand
- High peak power not available
- An ideal energy storage device:
  - Infinite shelf life
  - Negligible leakage
  - Unlimited capacity
  - Negligible volume
  - No need for energy conversion
  - Efficient energy acceptance and delivery

…Ideal battery doesn't exist

Energy Storage Options

	Li-Ion	**Thin Film Rechargeable**	**Super Cap**
Recharge Cycles	100s	5k-10k	Millions
Self Discharge	Moderate	Negligible	High
Charge Time	Hours	Minutes	Sec-Minutes
SMT & Reflow	Poor-None	Good	Poor
Physical Size	Large	Small	Medium
Capacity	0.3-2500mAHr	12-700uAHr	10-100uAHr
Environmental Impact	High	Minimal	Minimal

What is a Thin-Film Battery?

- Small, electrochemical batteries fabricated to deposit thin layers of battery materials
- Main Features:
  - Solid State Cell Chemistry
  - Superior Cycle Life
  - High Energy Density
  - Flexible packaging options
  - Negligible leakage
  - Rapid recharge
  - Broad temperature performance

Thin Film Battery Solutions

- **Cymbet**
  - Surface-mount
  - Packaged in QFN package
  - No harmful gases, liquids or special handling procedures
  - EnerChip CBC050 example
    - Output Voltage: 3.8V
    - Capacity: 50 µAh
    - Package: 16-pin M8 QFN
    - Size: 8 x 8 x 0.9 mm

- **Infinite Power Solutions**
  - Flexible, electrolyte based rechargeable lithium battery
  - Very thin: 0.11mm
  - Flexible
  - >10,000 recharge cycles
  - MEC101-7P example:
    - Output Voltage: 4.2V
    - Capacity: 700 µAh
    - Size: 25.4 x 25.4 x 0.11mm

EH System MCU Design Challenges

- Ability to operate with lowest standby current to maximize storage of energy
- Consume lowest possible power when active
- Ability to turn on and turn off instantaneously
- Efficient operation with lowest duty cycle of active vs. standby modes
- Analog capability for sensor interfacing and measurements
- Ability to operate with a low voltage range
- Lowest leakage currents to maximize harvested energy