

## Introduction

In this section we'll explore the ultra-low power abilities and architecture of the MSP430. We'll take a look at its low power modes and unique oscillator arrangement, along with techniques that can be used to minimize power consumption.

## Objectives

- Principles of ultra-low power applications
- Low power modes
- Oscillators
- Interrupts
- Ultra-low power lab

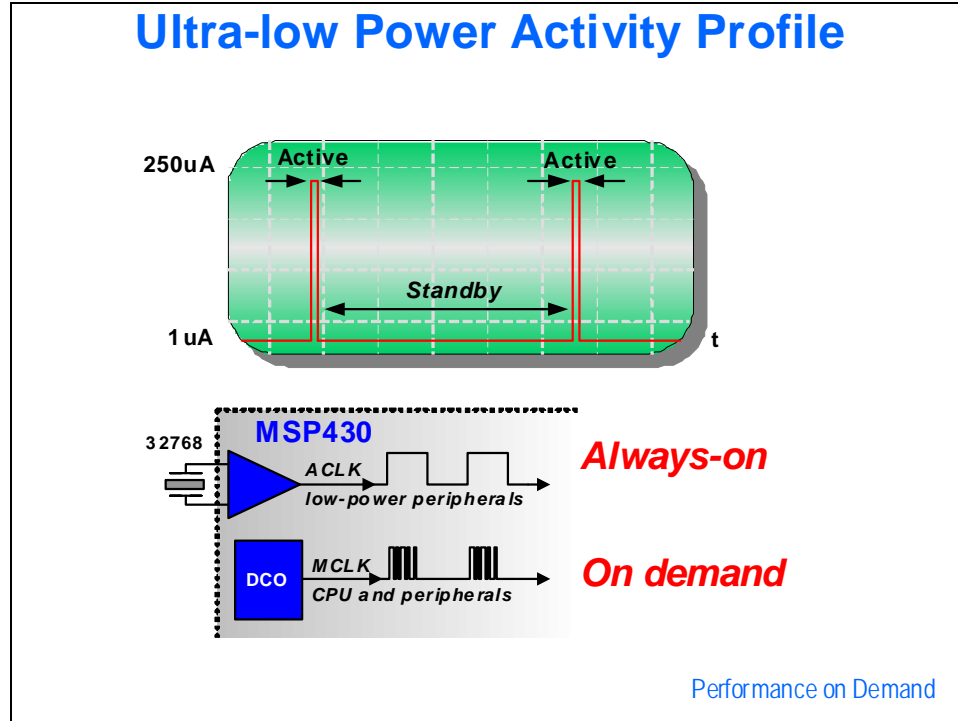
\*\*\* This page isn't really blank, you know. \*\*\*

# Module Topics

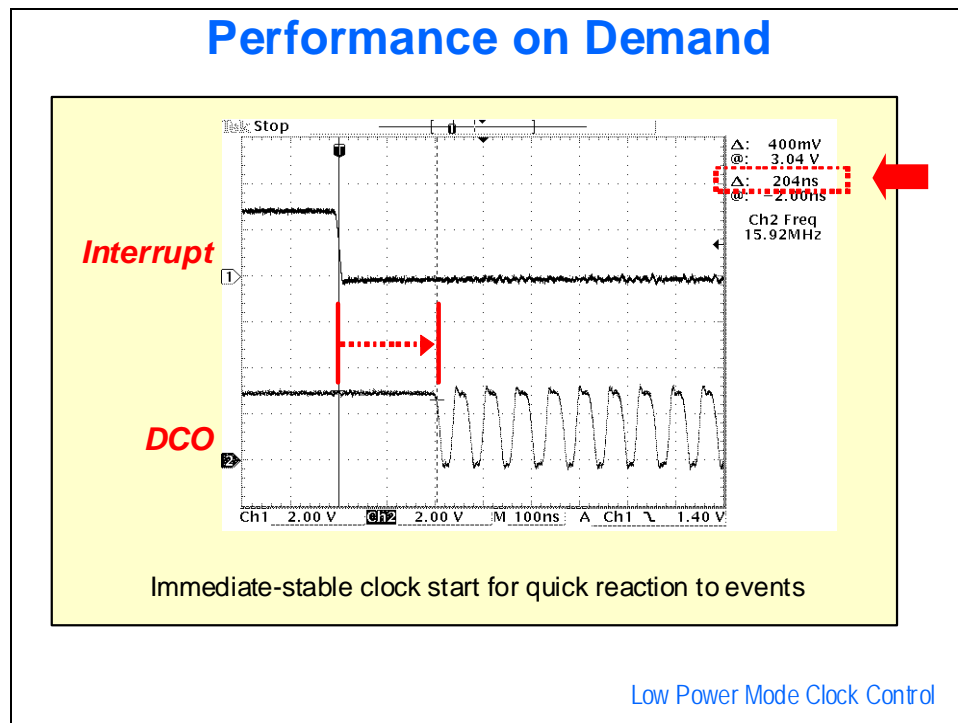
<b>Ultra-Low Power .....</b>	<b>2-1</b>
<i>Module Topics.....</i>	2-3
Activity Profile .....	2-5
Performance on Demand .....	2-5
Low Power Mode Clock Control.....	2-6
Low Power Mode Configuration .....	2-6
Low Power Modes in Assembly .....	2-7
Low Power Modes in C .....	2-7
11x/12x Basic Clock.....	2-8
1xx XTAL Options.....	2-8
1xx DCO Control .....	2-9
DCO Jitter.....	2-9
1xx DCO Calibration.....	2-10
2xx Basic Clock.....	2-10
4xx FLL.....	2-11
5xx Unified Clock System.....	2-11
5xx Power Management Module.....	2-12
Program Flow .....	2-12
Interrupt Processing.....	2-13
11x1 Interrupt Vectors.....	2-13
Move S/W Functions to Peripherals .....	2-14
Power Manage Internal Peripherals.....	2-14
Lowering System Power.....	2-15
Increasing Power Efficiency.....	2-15
Terminate Unused Pins.....	2-16
Ultra Low Power Principles.....	2-16
<i>Lab 3 – Ultra-Low Power in Practice.....</i>	2-17
Hardware list: .....	2-18
Software list:.....	2-18
<i>IAR Kickstart Procedure.....</i>	2-19
Lab3 Baseline .....	2-19
Lab3 using LPM3 .....	2-19
Shut Down .....	2-20
<i>Code Composer Studio 4.1 Procedure.....</i>	2-23
Lab3 Baseline .....	2-23
Lab3 using LPM3 .....	2-23
Shut Down.....	2-24
<i>Review Questions.....</i>	2-26

\*\*\* Let this be your doodle area \*\*\*

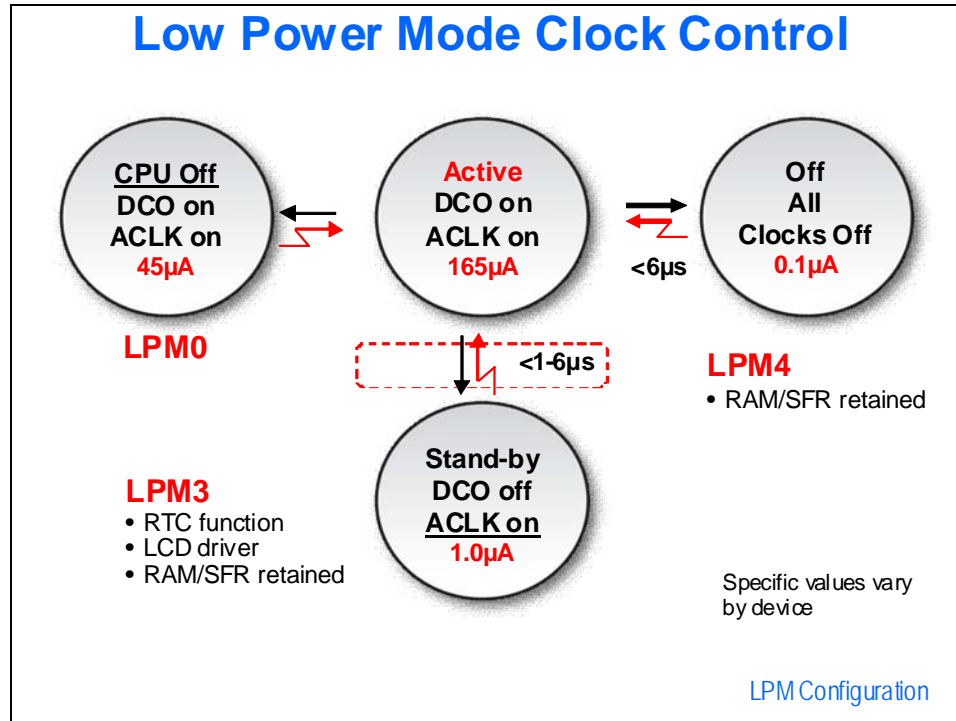
## Activity Profile



## Performance on Demand



## Low Power Mode Clock Control



## Low Power Mode Configuration

**Low Power Mode Configuration**

Reserved	V	SCG1	SCG0	OSC OFF	CPU OFF	GIE	N	Z	C
R2/SR									
Active Mode		0	0	0	0				~ 250µA
LPM0		0	0	0	1				~ 35µA
LPM3		1	1	0	1				~ 0.8µA
LPM4		1	1	1	1				~ 0.1µA

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

LPM in Assembly

## Low Power Modes in Assembly

### Low Power Modes In Assembly

```

ORG      0F000h
RESET    mov.w  #300h,SP
          mov.w
          #WDT_MDLY_32,&WDTCTL
          bis.b  #WDTIE,&IE1
          bis.b  #01h,&P1DIR

Mainloop bis.w  #CPUOFF+GIE,SR
          xor.b  #01h,&P1OUT
          jmp   Mainloop

WDT_ISR  bic.w  #CPUOFF,0(SP)
          reti

ORG      0FFFeh
DW       RESET
ORG      0FFF4h
DW       WDT_ISR
    
```

LPM in C

## Low Power Modes in C

### Low Power Modes In C

```

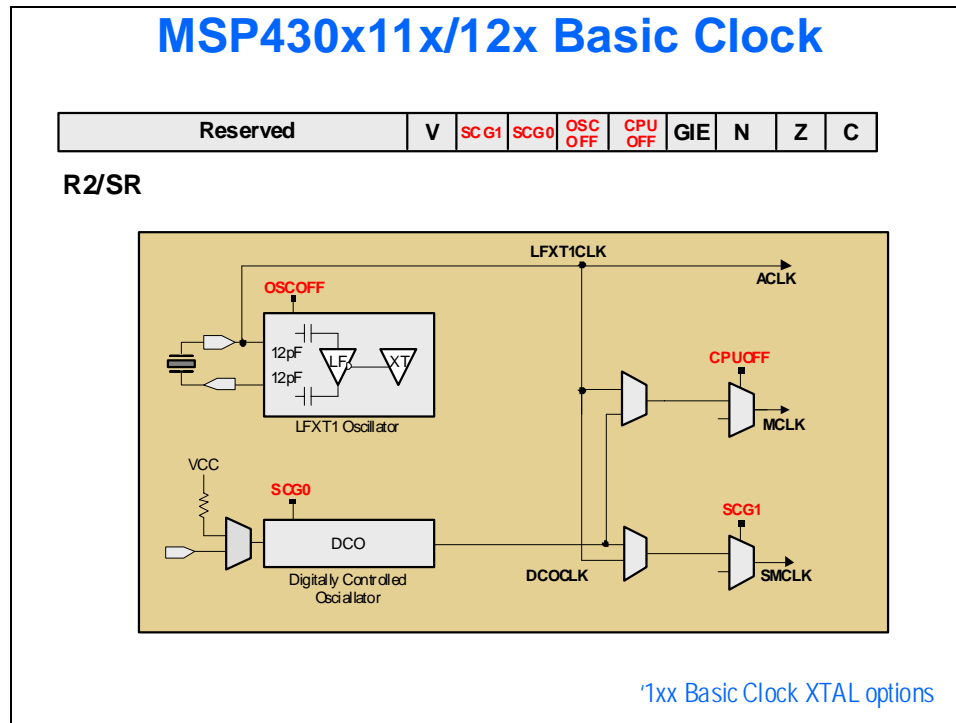
void main(void)
{
    WDTCTL = WDT_MDLY_32;
    IE1 |= WDTIE;
    P1DIR |= 0x01;

    for (;;)
    {
        _BIS_SR(CPUOFF + GIE);
        P1OUT ^= 0x01;
    }
}

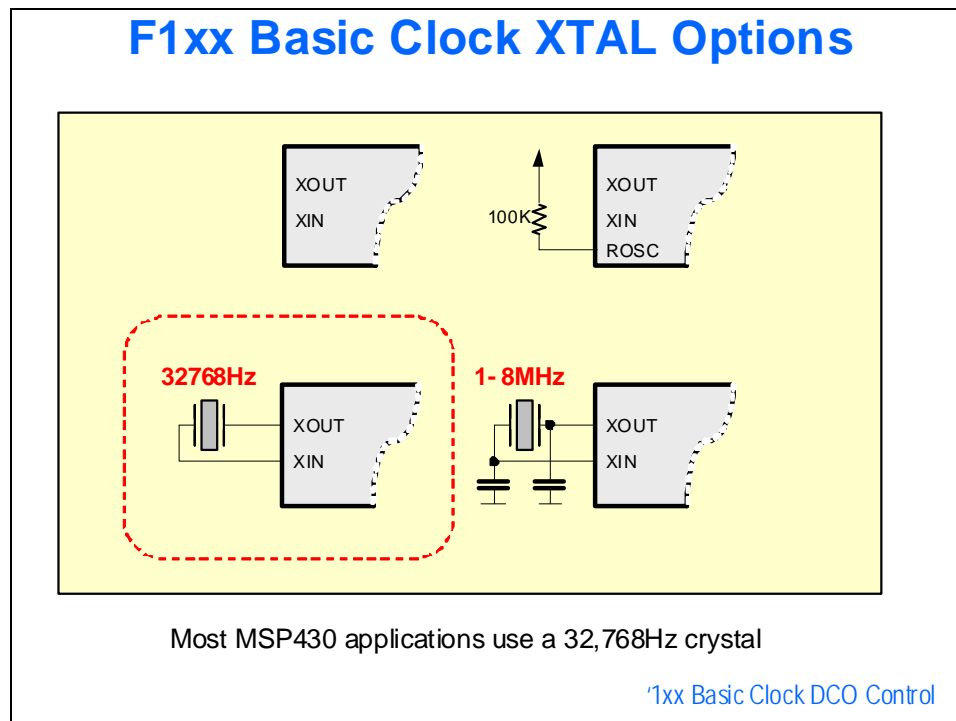
#pragma vector=WDT_VECTOR
__interrupt void watchdog_timer(void)
{
    BIC_SR_IRQ(CPUOFF);
}
    
```

'11x/'12x Basic Clock

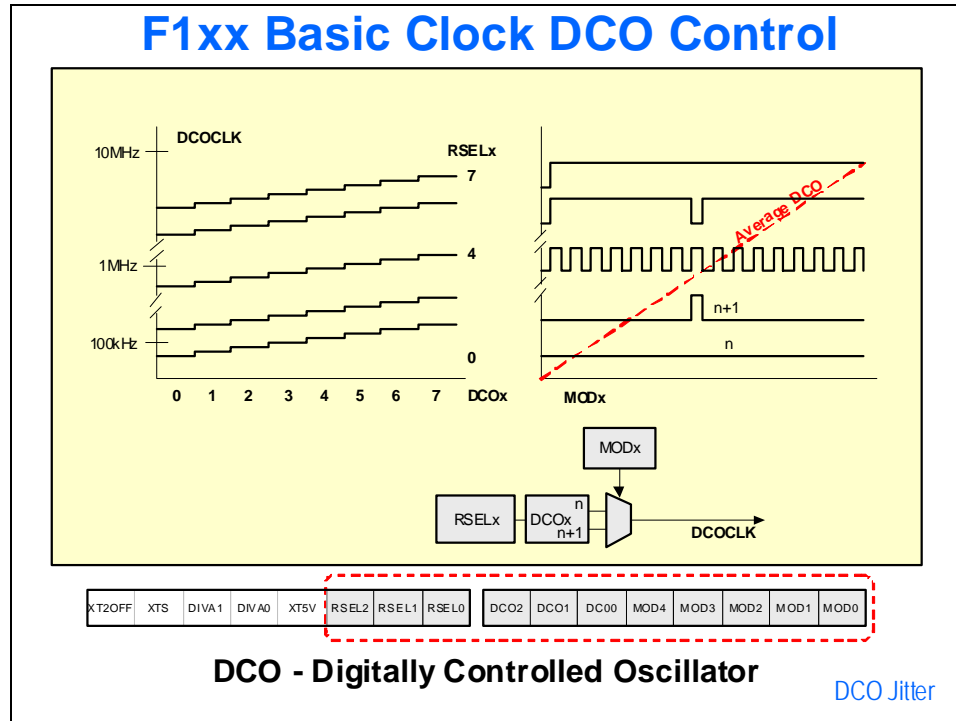
## 11x/12x Basic Clock



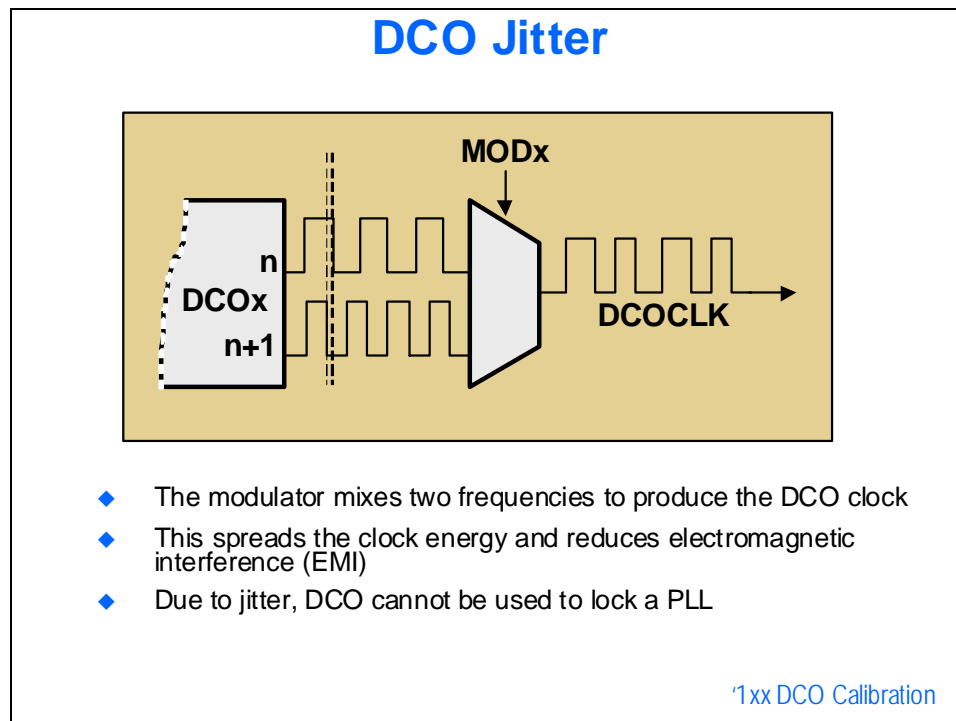
## 1xx XTAL Options



# 1xx DCO Control



# DCO Jitter



## 1xx DCO Calibration

### F1xx DCO Calibration

Clock precision is achieved by periodic adjustment

4096Hz  
ACLK

```
// Partial SW FLL Code
if (488 < Compare ) // DCO too fast
DCOCTL--;
else DCOCTL++; // DCO too slow
```

- ◆ Periodic loop adjusts DCO
- ◆ Known reference can be 50/60Hz AC power or 32kHz crystal frequency
- ◆ *If Rosc = 100k then DCOCLK ~ 2MHz*

F2xx Basic Clock +

## 2xx Basic Clock

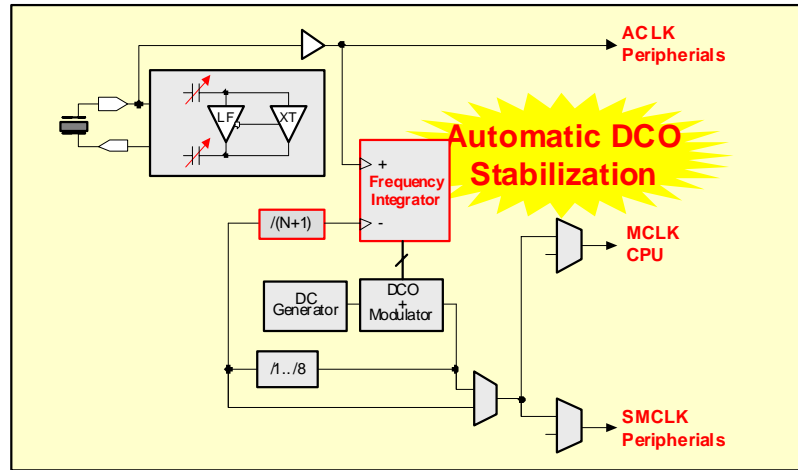
### F2xx Basic Clock+

- ◆ **LFXT1 XTAL Oscillator**
  - <1uA LPM3 standby mode
  - XTAL CAPs programmable
  - OSCfault LF/(XT)
  - Very Low Power Oscillator (VLO)
- ◆ **Improved DCO**
  - < 1us 0-to-16MHz
  - ± 2.5% DCO
  - Programmable frequency
- ◆ VLO not in F21x1

F4xx FLL

## 4xx FLL

### F4xx Frequency-Locked Loop (FLL)



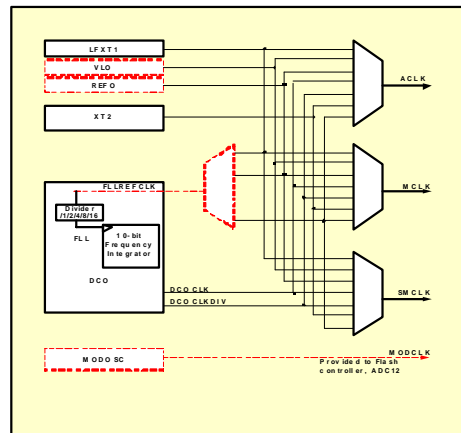
- ◆ Fully digital
- ◆ Oscillator fault fail-safe for LFXT1, DCO and XT2

F5xx UCS ...

## 5xx Unified Clock System

### F5xx: Unified Clock System

- ◆ **Orthogonal clock system**
  - ◆ Any source can drive any clock signal
- ◆ **2 Integrated clock sources:**
  - ◆ REFO: 32kHz, trimmed osc.
  - ◆ VLO: 12kHz, ultra-low power
- ◆ DCO & FLL provide high frequency accurate timing
- ◆ MODOSC provides bullet proof timing for Flash
- ◆ Crystal pins muxed with I/O function

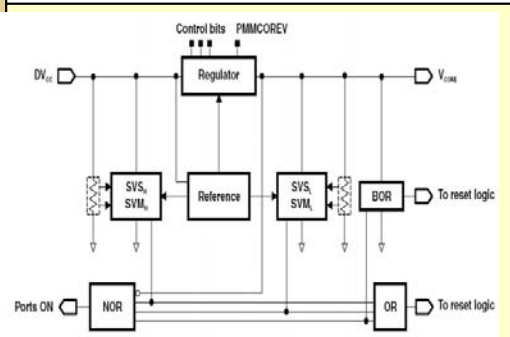


F5xx PMM ...

## 5xx Power Management Module

### F5xx: Power Management Module

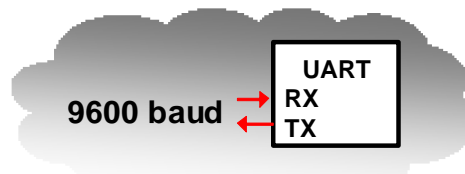
- ◆ Integrated LDO
- ◆  $V_{CORE}$  level programmable
- ◆ Flexibility in processing performance vs. power
- ◆ Integrated supervision & monitoring
- ◆ Zero-power BOR
- ◆ Five integrated supervisors
  - SVSH
  - SVSL
  - SVMH
  - SVML
  - BOR



Program flow ...

## Program Flow

### Interrupts Control Program Flow



```
// Polling UART Receive
for (;;)
{
    while (!(IFG2&URXIFG0));
    TXBUF0 = RXBUF0;
}
```

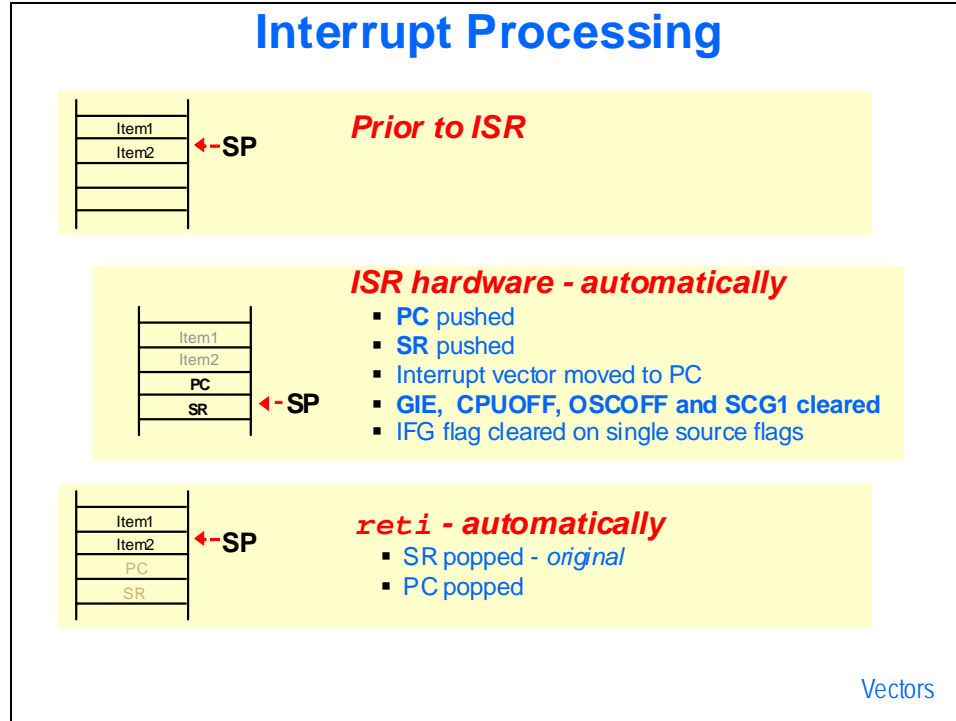
**100% CPU Load**

```
// UART Receive Interrupt
#pragma vector=UART_VECTOR
__interrupt void rx (void)
{
    TXBUF0 = RXBUF0;
}
```

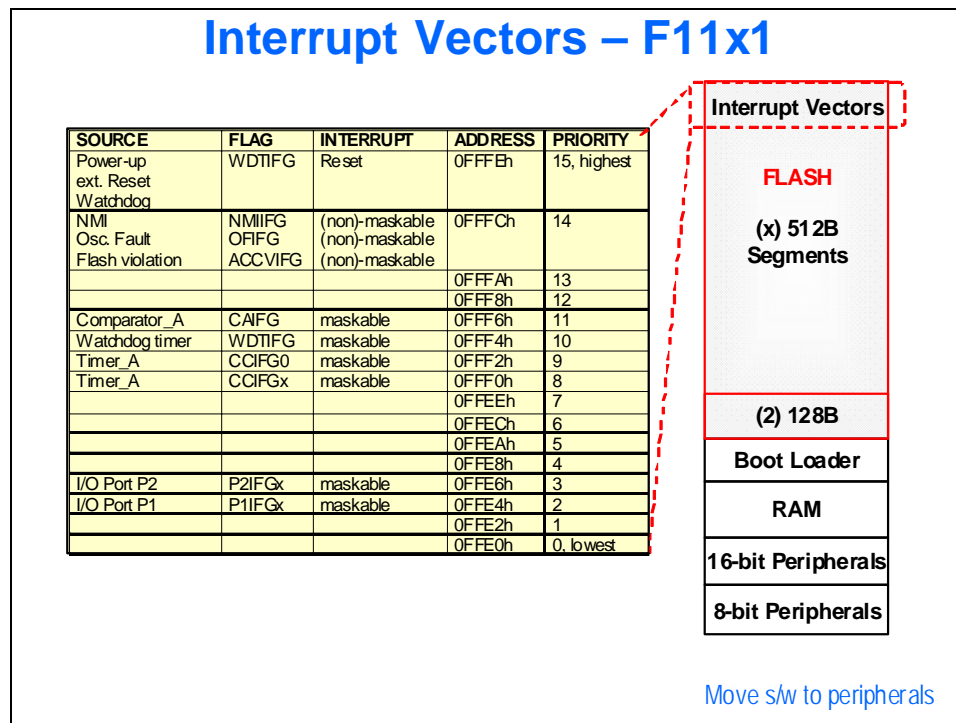
**0.1% CPU Load**

Interrupt Processing

# Interrupt Processing



# 11x1 Interrupt Vectors



## Move S/W Functions to Peripherals

### Move Software Functions to Peripherals

```
// Endless Loop
for (;;)
{
  P1OUT |= 0x04; // Set
  delay1();
  P1OUT &= ~0x04; // Reset
  delay2();
}
```

**100% CPU Load**

```
// Setup output unit
CCTL1 = OUTMOD0_1;
_BIS_SR(CPUOFF);
```

**Zero CPU Load**

Power manage internal peripherals

## Power Manage Internal Peripherals

### Power Manage Internal Peripherals

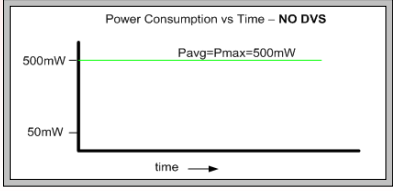
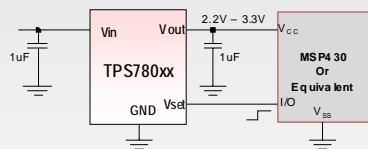
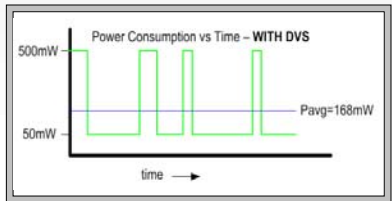
Comparator_A				
VCC	MIN	TYP	MAX	UNIT
2.2 V		25	40	μA
3 V		45	60	

```
P1OUT |= 0x02; // Power divider
CACTL1 = CARSEL + CAREF_2 + CAON; // Comp_A on
if (CAOUT & CACTL2)
  P1OUT |= 0x01; // Fault
else
  P1OUT &= ~0x01;
P1OUT &= ~0x02; // de-power divider
CACTL1 = 0; // Disable Comp_A
```

System power ...

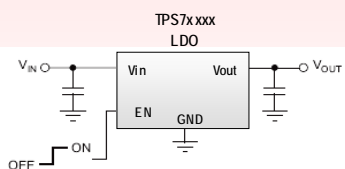
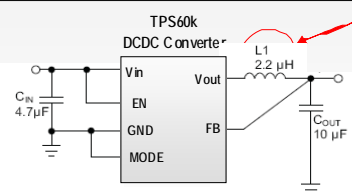
## Lowering System Power

### Lowering System Power

Problem	Solution
<p>My MCU is only operating a portion of the time, is there a way to lower the overall power consumption of my system?</p> 	<div style="text-align: center;">  </div> <p>Using a LDO with a programmable output voltage yields lower power consumption when your MCU is in an idle state</p> 
	<p>Power efficiency ...</p>

## Increasing Power Efficiency

### Increasing Power Efficiency

Problem	Solution
<p>I want higher efficiency than LDOs, but DCDC's are more complicated, right?</p> <div style="border: 1px solid #ccc; padding: 5px; margin: 10px 0;"> <p>Linear Regulator Efficiency = <math>V_{out}/V_{in}</math>                      - If <math>V_{in} = 5V, V_{out} = 3.3V \rightarrow \text{Eff} = 66\%</math></p> </div> 	<p>DCDC Converters with Integrated FETs have low external component count and dramatically reduce complexity with much greater efficiency</p> <div style="border: 1px solid #ccc; padding: 5px; margin: 10px 0;"> <p>Switching Regulator Efficiency = 85-95% typ                      - varies slightly with <math>V_{out}</math> &amp; output current                      - one additional component vs LDO solution</p> </div> 
<div style="background-color: #333; color: white; padding: 10px; border-radius: 5px;"> <p>Translating to Real World Applications:                      If LDO eff = 66% &amp; DCDC eff = 90% : DCDC allows your battery to last ~36% longer</p> </div>	
	<p>Unused pins ...</p>

## Terminate Unused Pins

### Terminate Unused Pins

- ◆ Unused port pins Px.0 – Px.7
  - ◆ Set as output direction to avoid floating gate current
- ◆ XT2IN, XT2OUT?
- ◆ See the last page of chapter 2 in the user's guide

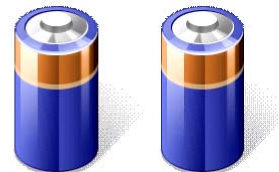


Low-power principals ...

## Ultra Low Power Principles

### Principles For ULP Applications

- ◆ Maximize the time in LPM3
- ◆ Use interrupts to control program flow
- ◆ Replace software with peripherals
- ◆ Power manage external devices
- ◆ Configure unused pins properly
- ◆ Efficient code makes a difference
- ◆ Even wall powered devices can be “greener”
- ◆ **Every unnecessary instruction executed is a portion of the battery wasted that will never return**



Lab3 ...

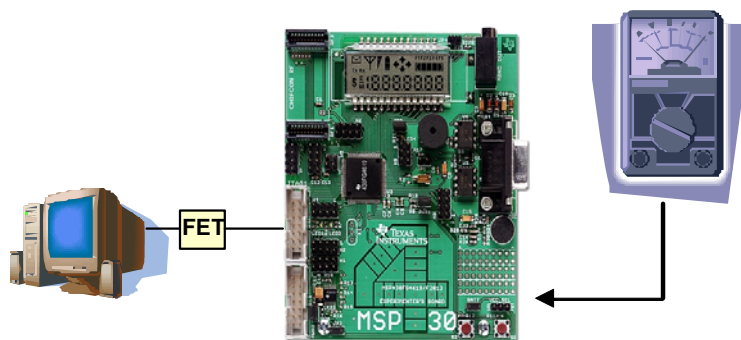
## Lab 3 – Ultra-Low Power in Practice

We're going to measure the power saving effect of using LPM3 mode.

### Lab3: Ultra Low-Power In Practice

The code from lab 2 has been converted to use LPM3 instead of the while(1) loop

Using an ammeter, measure the current through the PWR1 jumper



[Review](#)

## Hardware list:

- WinXP PC
- MSP-FET430UIF
- USB cable
- JTAG ribbon cable
- MSP430FG461x/F28xx Experimenter's Board with batteries
- Digital Multimeter
- Jumpers
- Two AAA Batteries

## Software list:

- IAR Kickstart for MSP430 version 4.21B
- Code Composer Studio 4.1
- Labs
- Additional pdf documentation
- Adobe™ Reader

# IAR Kickstart Procedure

The C code from the previous lab has been modified to use LPM3 mode instead of the while(1) loop. We'll measure the current draw of both labs.

## Lab3 Baseline

### 1. Set up the Hardware

**Remove** the PWR1 jumper from the Experimenter's Board and **place** it over a nearby pin so you won't lose it.

Make sure the two AAA batteries are in place and connect the **BATT** jumper to power the board.

**Hook up** the positive lead of the multimeter to the **right-hand PWR1** pin and the negative lead to the **left-hand PWR1** pin. Make sure the leads are connected to the proper jacks on the multimeter. Place the multimeter in the **lowest** milliamp measurement setting and **turn it on**.

### 2. Run the Software

Your **Lab2** software should still be loaded in the F4618/9 (as well as the Lab1 code in the F2013). If for some reason the Lab2 code is not running, use the steps in Lab2 to reload and run it. **Remove** the JTAG cable from the FG4618/9 debug port. You may have to **remove and replace** the BATT jumper to get the MSP430 to boot properly. **Press S1** a couple times to verify that the software is functioning.

### 3. Measure the current

**Fill in** the blanks in the chart below for Lab2 with LED3 on and off.

Code used	LED Off (mA)	LED On (mA)
Lab2		
Lab3		

## Lab3 using LPM3

### 4. Start up IAR Kickstart

Start up *IAR Kickstart*. When prompted, load the Lab2 workspace. The Lab2 code is probably visible in the editor window. Close the editor by clicking the tiny, little **X** in the upper right-hand corner of the editor window (not the one that closes *IAR Embedded Workbench*).

### 5. Swap out the Source Files

**Right-click** on **Lab2\_exercise.c** in the Workspace window and select **Remove**. When prompted whether or not you are sure, click **Yes**.

On the menu bar, click **Project** ⇒ **Add Files**. Navigate to **C:\MSP430\IAR Labs\Lab3** and select **Lab3\_solution.c**. Click **Open**.

## 6. Inspect the Modified Code

Double-click on **Lab3\_solution.c** in the Workspace window to open it in the editor. Take a moment to inspect the Lab3\_solution code. Note the configuration of unused pins in the initialization as well as the use of LPM3 in the while(1) loop. The while(1) loop itself has been altered somewhat to decrease power. Note also the ISR code changes.

## 7. Build, Download and Run

**Replace** the JTAG cable in the FG4618/9 debug port. Click the **Debug** button to build and download the code to the Experimenter's Board. Correct any errors you may find. When you've successfully downloaded the code to the board, Click the **Stop Debugging** button in *IAR Embedded Workbench* and **remove** the JTAG cable from the FG4618/9 debug port. You may have to **remove and replace** the BATT jumper to get the MSP430 to boot properly. **Press S1** a couple times to verify that the software is functioning

## 8. Measure the Lab3 current

**Fill in** the remaining cells in the table in step 3.

## 9. Analysis

We made the same measurements, and here's what we got:

Code used	LED Off (mA)	LED On (mA)
Lab2	0.6	2.7
Lab3	0.0	2.1

Obviously, the current for Lab3 with the LED off was below the measurement abilities of the meter we were using. Subsequent measurements with a better (more expensive) multimeter showed that the current was 1.5uA. That's a current reduction of about 97%.

## Shut Down

### 10. Shut Down

**Turn off** the multimeter and remove the leads from the PWR1 pins. **Replace** the PWR1 jumper.

**Remove** the BATT jumper and **place** it over one pin for safekeeping.

**Shut down** *IAR Embedded Workbench*. When prompted to save the project, click **No**.

**Replace** the JTAG cable in the FG4618/9 debug port.

## 11. Some further questions

**Why were the I/Os configured as they were?**

**Why was LPM3 used?**

**Look in the header file to see how LPM3\_bits is defined**

**What further low-power improvements could be made?**

You can find the answers to these questions in the Addendum section at the end of this workbook.



IAR Kickstart users ... You're done.  
Proceed to the review questions on page 2-26.

\*\*\* Where is my flying car? \*\*\*

# Code Composer Studio 4.1 Procedure

The C code from the previous lab has been modified to use LPM3 mode instead of the while(1) loop. We'll measure the current draw of both labs.

## Lab3 Baseline

### 1. Set up the Hardware

**Remove** the PWR1 jumper from the Experimenter's Board and **place** it over a nearby pin so you won't lose it.

Make sure the two AAA batteries are in place and connect the **BATT** jumper to power the board.

**Hook up** the positive lead of the multimeter to the **right-hand PWR1** pin and the negative lead to the **left-hand PWR1** pin. Make sure the leads are connected to the proper jacks on the multimeter. Place the multimeter in the **lowest** milliamp measurement setting and **turn it on**.

### 2. Run the Software

Your **Lab2** software should still be loaded in the F4618/9 (as well as the Lab1 code in the F2013). If for some reason the Lab2 code is not running, use the steps in Lab2 to reload and run it. **Remove** the JTAG cable from the FG4618/9 debug port. You may have to **remove and replace** the BATT jumper to get the MSP430 to boot properly. **Press S1** a couple times to verify that the software is functioning.

### 3. Measure the current

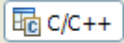
**Fill in** the blanks in the chart below for Lab2 with LED3 on and off.

Code used	LED Off (mA)	LED On (mA)
Lab2		
Lab3		

## Lab3 using LPM3

### 4. Start up CCS

Start up CCS. When prompted, select the **Lab2** workspace. If CCS opens in the debug

perspective, click on  on the upper right of the menu bar to return to the editing perspective. The Lab2 code is probably visible in the editor window. Close the editor window by clicking the **X** in the **Lab2\_exercise.c** tab.

## 5. Swap out the Source Files


**Right-click** on **Lab2\_exercise.c** in the Project pane and select **Delete**. When prompted whether or not you are sure, click **Yes**.


On the menu bar, click **Project** ⇒ **Add Files to Active Project ...** Navigate to **C:\MSP430\CCS Labs\Lab3** and select **Lab3\_solution.c**. Click **Open**.

## 6. Inspect the Modified Code

Double-click on **Lab3\_solution.c** in the Project pane to open it in the editor. Take a moment to inspect the **Lab3\_solution** code. Note the configuration of unused pins in the initialization as well as the use of **LPM3** in the **while(1)** loop. The **while(1)** loop itself has been altered somewhat to decrease power. Note also the **ISR** code changes.

## 7. Build, Download and Run

Make sure that the **JTAG** cable in the **FG4618/9** debug port. Click the  **Debug Launch** button to build and download the code to the Experimenter's Board. Correct any errors you may find.

When you've successfully downloaded the code to the board, Click the **Terminate All**  button in **CCS** and **remove** the **JTAG** cable from the **FG4618/9** debug port. You may have to **remove and replace** the **BATT** jumper to get the **MSP430** to boot properly. **Press S1** a couple times to verify that the software is functioning

## 8. Measure the Lab3 current

**Fill in** the remaining cells in the table in step 3.

## 9. Analysis

We made the same measurements, and here's what we got:

Code used	LED Off (mA)	LED On (mA)
<b>Lab2</b>	<b>0.6</b>	<b>2.7</b>
<b>Lab3</b>	<b>0.0</b>	<b>2.1</b>

Obviously, the current for **Lab3** with the **LED** off was below the measurement abilities of the meter we were using. Subsequent measurements with a better (more expensive) multimeter showed that the current was **1.5uA**. That's a current reduction of about **97%**.

## Shut Down

### 10. Shut Down

**Turn off** the multimeter and remove the leads from the **PWR1** pins. **Replace** the **PWR1** jumper.

**Remove** the **BATT** jumper and **place** it over one pin for safekeeping.

**Shut down** *Code Composer Studio*.

**Replace** the **JTAG** cable in the **FG4618/9** debug port.

## 11. Some further questions

**Why were the I/Os configured as they were?**

**Why was LPM3 used?**

**Look in the header file to see how LPM3\_bits is defined**

**What further low-power improvements could be made?**

You can find the answers to these questions in the Addendum section at the end of this workbook.



You're done

## Review Questions

### Review

- ◆ To minimize power consumption, you should maximize your time in what LPM mode?
- ◆ Why are unused pins set as outputs?
- ◆ You should control program flow with ...
- ◆ Most MSP430 designs utilize a \_\_\_\_\_ crystal.

You can find the answers to these questions in the Addendum section at the end of this workbook.