**Introduction**

- Power-efficient MSP430 apps:
  - Minimize instantaneous current draw
  - Maximize time spent in low-power modes
- The MSP430 is inherently low-power
- But your design has a big impact on power efficiency!
- Proper low-power design techniques make the difference
Agenda

- Introduction
- The Basics: Operating Modes & Clocks
  - CPU and Operating Conditions
  - Optimizing Peripheral Modes
  - Delegating Functionality to Peripherals
  - Software & Cycle Reduction

Introduction: Ultra-Low-Power Architecture

Multiple operating modes
- 0.1uA power down
- 0.7uA standby mode
- 250uA / 1MIPS

Zero-power BOR

Modern CPU
Minimum cycles per task

50nA pin leakage

Instant-on stable high-speed clock

Intelligent peripherals

ATC 2008
MSP430 Advanced Technical Conference

Texas Instruments
Operating Modes

CPU Off
DCO on
ACLK on
45-65uA

Active
DCO on
ACLK on
300-500uA

Off
All
Clocks Off
0.1uA

LPM0
LPM3
· RTC function
· LCD driver
· RAM/SFR retained

Stand-by
DCO off
ACLK on
1.0uA

LPM4
· RAM/SFR retained

Specific values vary by device

Clock Management: 1xx, 2xx, 4xx

<table>
<thead>
<tr>
<th>R2/SR:</th>
<th>Reserved</th>
<th>V</th>
<th>SCG1</th>
<th>SCG0</th>
<th>OSC OFF</th>
<th>CPU OFF</th>
<th>GIE</th>
<th>N</th>
<th>Z</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always-on</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Active Mode</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LPM3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

On demand

OSCOFF

12pF

12pF

LFXT1 Oscillator

SCG0

DCO

Digitally Controlled Oscillator

LFXT1CLK

MCLK

OSC

VCC

CPUOFF

SMCLK

ALWAYS-ON

ACTIVEMODE

LPM3
**Instant-On Clock**

- Instant response to events, even from sleep modes
- As a result, MSP430 can spend more time asleep!

**Using MSP430 Operating Modes**

- Maximize time spent in low-power modes
  - Set up interrupt handling and then go to sleep
- Use ACLK for peripherals
  - Allows use of LPM3 instead of LPM0
Agenda

- Introduction
- The Basics: Operating Modes & Clocks
- CPU and Operating Conditions
  - Optimizing Peripheral Modes
  - Delegating Functionality to Peripherals
  - Software & Cycle Reduction

Effects of Vcc / MCLK / Temperature

- Power draw increases with...
  - Vcc
  - CPU clock speed (MCLK)
  - Temperature
- Slowing MCLK reduces instantaneous power, but usually increases active duty cycle
  - Power savings nullified – best to use default MCLK (or increase it if required for application performance)
- 5xx has integrated LDO with variable output voltage
  - Optimize core voltage for chosen MCLK speed
**Vcc / MCLK / Temp: Typical Relationships**

![Graph showing active mode current vs Vcc, T_A = 25°C](image)

![Graph showing active mode current vs DCO frequency](image)

**Power Consumption in CMOS Designs**

![Diagram of CMOS inverter](image)

When the P- and N-channel devices are on simultaneously, the result is "shoot-through" current
MCU’s Digital Supply Current

![Diagram of MCU's digital supply current]

**ATC 2008**
MSP430 Advanced Technical Conference

### Configuring Unused I/Os
- Digital input pins subject to the same effect
  - Input voltages between $V_{IL}$ and $V_{IH}$ cause shoot-through
- This can occur if input is allowed to “float” (unconnected)
- Port I/Os should be...
  - Driven as outputs
  - Be driven at Vcc/ground by an external device
  - Have a pull-up/down resistor
Agenda

- Introduction
- The Basics: Operating Modes & Clocks
- CPU and Operating Conditions

- Optimizing Peripheral Modes
  - Delegating Functionality to Peripherals
  - Software & Cycle Reduction

**Power-Manage Internal Peripherals**

Only activate peripherals while used, disable when finished

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

<table>
<thead>
<tr>
<th>OAPMx</th>
<th>Icc (uA, typ)</th>
<th>Slew Rate (V/us)</th>
<th>GBW* (Mhz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>50</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>10</td>
<td>110</td>
<td>0.8</td>
<td>1.4</td>
</tr>
<tr>
<td>11</td>
<td>180</td>
<td>1.2</td>
<td>2.2</td>
</tr>
</tbody>
</table>

*Non-inverting.
MSP430FG4619
OARRIP=0. OARRIP has a different function in the 2xx family.

Look for opportunities to save power by optimizing performance tradeoffs for your application

Power Management Options: DAC12

<table>
<thead>
<tr>
<th>DAC12AMPx</th>
<th>Idd (uA, typ)</th>
<th>BW_3dB(kHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>010</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>200</td>
<td>180</td>
</tr>
<tr>
<td>100</td>
<td>700</td>
<td>550</td>
</tr>
<tr>
<td>110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>111</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MSP430FG4619
Power Manage External Devices

- External op amp with shutdown can be 20x lower total power

Clock Management: Clock Requests

- Module can use a clock request to force its source to stay active, even when entering LPMx
  - LPMx otherwise goes into effect
  - When clock request goes away, clock shuts down & LPMx fully implemented
- Helps module respond to events without CPU intervention
- If code uses “wrong” LPM mode, may see higher current draw than you expect
 Agenda

• Introduction
• The Basics: Operating Modes & Clocks
• CPU and Operating Conditions
• Optimizing Peripheral Modes
• Delegating Functionality to Peripherals
• Software & Cycle Reduction

 Moving Functionality to Peripherals

• Peripherals use less current than CPU
• Delegating to them allows CPU to shut down, saving system power
• “Intelligent” peripherals are more capable, providing more opportunity for CPU shutoff
Intelligent Peripherals: Data Converters

- ADC10/ADC12_A oscillator & core are enabled/disabled with conversion activity (automatic)
- Internal reference control
  - Band Gap
  - Buffer
- Sequencing of channels can be automated
- Data transfer can be automated with DMA or DTC
- Timer triggering available on ADC10, 12, 12_A

ADC10

- 10-bit ADC
- 200ksps+
- Autoscan
- Single Sequence
  - Repeat-single
  - Repeat-sequence
- Internal/external reference
- TA SOC triggers
- Direct transfer controller (DTC)
Why Is Autoscan + DTC Important?

// Software
RES[pRes++] = ADC10MEM;
ADC10CTL0 &= ~ENC;
if (pRes < NR_CONV)
{
    CurrINCH++;
    if (CurrINCH == 3)
        CurrINCH = 0;
    ADC10CTL1 &= ~INCH_3;
    ADC10CTL1 |= CurrINCH;
    ADC10CTL0 |= ENC+ADC10SC;
}
// Autoscan + DTC
_BIS_SR(CPUOFF);

Is Timer-Triggered ADC Important?

// Interrupt
; MSP430 ISR to start conversion
BIS #ADC12SC,&ADC12CTL0 ; Start conversion
RETI ; Return

ATC 2008
MSP430 Advanced Technical Conference
Intelligent peripherals: DMA

- DMA enabled for repetitive data handling
- Minimal software requirements and CPU cycles

**ATC 2008**
MSP430 Advanced Technical Conference

### Agenda
- Introduction
- The Basics: Operating Modes & Clocks
- CPU and Operating Conditions
- Optimizing Peripheral Modes
- Delegating Functionality to Peripherals
- Software & Cycle Reduction

**ATC 2008**
MSP430 Advanced Technical Conference
Reduce Cycles

- CPU active time is a direct function of how many cycles need to be executed
- Reducing cycles is key to maximizing the use of low-power modes
- Many ways to do this, but an important one is interrupt-driven coding

Interrupts Control Program Flow

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

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

9600 baud

100% CPU Load

0.1% CPU Load
Interrupt Vector Generator Registers

- IVGs efficiently decode interrupt vectors & allow more sources
- When flag is set, IVG contains an offset address according to the kind of event that occurred
- Add offset to the PC and use a jump-table
- If multiple events, IVG contains the highest-priority pending interrupt
- Using IVGs instead of flag polling greatly reduces interrupt overhead

Bytes, Words & CPU Registers

- Use CPU registers for calculations & dedicated variables
- Same code size for word or byte
- Use word operations when possible
Effect Of The Constant Generator

- Immediate values 0xFFFF, 0, 1, 2, 4, 8 generated in hardware
- Reduces code size and cycles
- Completely automatic!

C coding tips

- Use local variable as much as possible. Local variables use CPU registers whereas global variables use RAM.
- Use bit mask instead of bit fields for unsigned int and unsigned char.
- Use unsigned data types where possible
- Use pointers to access structures and unions
- Use “static const” class to avoid run-time copying of structures, unions, and arrays.
- Avoid modulo
- Count down “for” loops
- Get to know your C code and its disassembly!
Summary

• Proper system design is necessary for the best low-power performance
• Maximize power efficiency by minimizing program duty cycle
• Make good use of all power modes
• Reduce program cycles

Thank you.