Agenda

• Intro to MSP430 Architecture
• Intro to Tools and I/O
• Intro to Low-Power with the MSP430
TI Microcontroller Portfolio

- **High-Performance**
  - Motor Control
  - Digital Power Supply

- **Industry Standard**
  - Automotive
  - General Purpose

- **Measurement**
  - Utility Metering
  - Portable Instrumentation

- **MSP430 Ultra-low Power**
- **TMS470 ARM7TDMI**
- **C2000 150 MIPS**

$25B MICRO TAM

16/32-bit

8-bit

Performance

DSP
MSP430 Products

Device
- Production
- Sampling
- Development
- Future
- Signal Chain on Chip

Performance

5xx
- 25 MIPS+
- 128-256 KB
- USB-Zigbee™

2xx
- 16 MIPS
- 1-120KB
- 500 nA
- Stand By

1xx
- 8 MIPS
- 1-60KB

F = Flash
C = Custom ROM

Integration

4xx-ASSP
- 8 MIPS
- 4-120KB
- LCD Driver

© 2006 Texas Instruments Inc, Slide 4
What Is The MSP430?

**Ultra-low Power**
- 0.1μA power down
- 0.8μA standby mode
- 250μA / 1MIPS
- <1μs clock start-up
- <50nA port leakage
- **Zero-power** BOR

**Ultra-Flexible**
- 1k-128kB ISP Flash
- 14-100 pin options
- USART, I2C, Timers
- 10/12/16-bit ADC
- DAC, OP Amp, LCD driver
- **Embedded emulation**
Orthogonal Architecture

- Clear and consistent with no exceptions
- Compiler efficient

- Special instructions to learn
- Complicated and inefficient
16-bit RISC CPU

- Single-cycle register file
  - 4 special purpose
  - 12 general purpose
  - No accumulator bottleneck

- RISC architecture
  - 27 core instructions
  - 24 emulated instructions
  - 7 addressing modes

- Memory-to-memory atomic addressing

- Bit, byte and word processing
Deep Register File RISC Advantage

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov</td>
<td>#16,R15     ; Loop Counter</td>
</tr>
<tr>
<td>clr</td>
<td>R14         ; 0 -&gt; RESULT MSD</td>
</tr>
<tr>
<td>clr</td>
<td>R13         ; 0 -&gt; RESULT LSD</td>
</tr>
<tr>
<td>rla</td>
<td>R12         ; Binary MSB to carry</td>
</tr>
<tr>
<td>dadd</td>
<td>R13,R13     ; RESULT x2 LSD</td>
</tr>
<tr>
<td>dadd</td>
<td>R14,R14     ; MSD</td>
</tr>
<tr>
<td>dec</td>
<td>R15         ; Through?</td>
</tr>
<tr>
<td>jnz</td>
<td>BIN1        ; Not through</td>
</tr>
</tbody>
</table>

- Access to many single-cycle registers
- Useful for calculation intensive functions
Bytes, Words And CPU Registers

; 16-bit addition
5405   add.w  R4,R5         ; 1/1
529202000202 add.w  &0200,&0202 ; 3/6

; 8-bit addition
5445   add.b  R4,R5         ; 1/1
52D202000202 add.b  &0200,&0202 ; 3/6

• Use CPU registers for calculations and dedicated variables
• Same code size for word or byte
• Use word operations when possible

© 2006 Texas Instruments Inc, Slide 9
Do You Use A 10+ Bit ADC?

84 bits / 24 cycles

48 bits / 6 cycles

; Other MCU
movf ADCRESH,W
movwf RAMH
bsf STATUS,0x20
movf ADCRESL,W
bcf STATUS,0x20
movwf RAML

; MSP430
mov ADC10MEM, RAM

© 2006 Texas Instruments Inc, Slide 10
# Seven Addressing Modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Instruction</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register Mode</td>
<td>mov.w R10,R11</td>
<td>Single cycle</td>
</tr>
<tr>
<td>Indexed Mode</td>
<td>mov.w 2(R5),6(R6)</td>
<td>Table processing</td>
</tr>
<tr>
<td>Symbolic Mode</td>
<td>mov.w EDE,TONI</td>
<td>Easy to read code, PC relative</td>
</tr>
<tr>
<td>Absolute Mode</td>
<td>mov.w &amp;EDE,&amp;TONI</td>
<td>Directly access any memory location</td>
</tr>
<tr>
<td>Indirect Register Mode</td>
<td>mov.w @R10,0(R11)</td>
<td>Access memory with pointers</td>
</tr>
<tr>
<td>Indirect Autoincrement</td>
<td>mov.w @R10+,0(R11)</td>
<td>Table processing</td>
</tr>
<tr>
<td>Immediate Mode</td>
<td>mov.w #45h,&amp;TONI</td>
<td>Unrestricted constant values</td>
</tr>
</tbody>
</table>

Atomic
Atomic Addressing

\[ B = B + A \]

- Non-interruptible memory-to-memory operations
- Useable with complete instruction set

; Pure RISC
push R5
ld R5,A
add R5,B
st B,R5
pop R5

; MSP430
add A,B

© 2006 Texas Instruments Inc, Slide 12
Write Less Code

; Other MCU
;   movlw HIGH Tab
movwf PCLATH
movwf DispVal,W
call Tab
movwf PORTB
goto Continue
Tab addwf PCL,F
retlw B'00111111'
retlw B'00000110'
retlw B'01011011'
retlw B'01001111'
retlw B'01100110'
retlw B'01101101'
retlw B'01111101'
retlw B'00000111'
retlw B'01111111'
retlw B'01101111'
Continue

; MSP430
;   mov.b Tab(DispVal),P1OUT
Tab DW 0063Fh
DW 04F5Bh
DW 06E66h
DW 0077Ch
DW 0677Fh

238 bits / 48 cycles

128 bits / 6 cycles
Effect Of The Constant Generator

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

```
4314  mov.w #0002h,R4     ; With CG
40341234  mov.w #1234h,R4 ; Without CG
```

Completely Automatic!
24 Emulated Instructions

- Easier to understand - no code size or speed penalty
- Replaced by assembler with core instructions

4130  ret  ; Return (emulated)
4130  mov.w  @SP+,PC  ; Core instruction

Completely Automatic!
# Three Instruction Formats

<table>
<thead>
<tr>
<th>Op-Code</th>
<th>S-Register</th>
<th>Ad</th>
<th>B/W</th>
<th>As</th>
<th>D-Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>; Format I Source and Destination</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5405</td>
<td>add.w</td>
<td>R4,R5</td>
<td></td>
<td></td>
<td>; R4+R5=R5 xxxx</td>
</tr>
<tr>
<td>5445</td>
<td>add.b</td>
<td>R4,R5</td>
<td></td>
<td></td>
<td>; R4+R5=R5 00xx</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Op-Code</th>
<th>B/W</th>
<th>As</th>
<th>D/S-Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>; Format II Destination Only</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6404</td>
<td></td>
<td></td>
<td>rlc.w R4</td>
</tr>
<tr>
<td>6444</td>
<td></td>
<td></td>
<td>rlc.b R4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Op-Code</th>
<th>Condition</th>
<th>10-bit PC offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>; Format III There are 8 (Un)conditional Jumps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3C28</td>
<td>jmp</td>
<td>Loop_1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; Goto Loop_1</td>
</tr>
</tbody>
</table>
## 51 Total Instructions

<table>
<thead>
<tr>
<th>Format I Source, Destination</th>
<th>Format II Single Operand</th>
<th>Format III +/- 9bit Offset</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>add(.b)</code></td>
<td><code>br</code></td>
<td><code>jmp</code></td>
<td><code>clrc</code></td>
</tr>
<tr>
<td><code>addc(.b)</code></td>
<td><code>call</code></td>
<td><code>jc</code></td>
<td><code>setc</code></td>
</tr>
<tr>
<td><code>and(.b)</code></td>
<td><code>swpb</code></td>
<td><code>jnc</code></td>
<td><code>clrz</code></td>
</tr>
<tr>
<td><code>bic(.b)</code></td>
<td><code>sxt</code></td>
<td><code>jeq</code></td>
<td><code>setz</code></td>
</tr>
<tr>
<td><code>bis(.b)</code></td>
<td><code>push(.b)</code></td>
<td><code>jne</code></td>
<td><code>clrn</code></td>
</tr>
<tr>
<td><code>bit(.b)</code></td>
<td><code>pop(.b)</code></td>
<td><code>jge</code></td>
<td><code>setn</code></td>
</tr>
<tr>
<td><code>cmp(.b)</code></td>
<td><code>rra(.b)</code></td>
<td><code>jl</code></td>
<td><code>dint</code></td>
</tr>
<tr>
<td><code>dadd(.b)</code></td>
<td><code>rrc(.b)</code></td>
<td><code>jn</code></td>
<td><code>eint</code></td>
</tr>
<tr>
<td><code>mov(.b)</code></td>
<td><code>inv(.b)</code></td>
<td></td>
<td><code>nop</code></td>
</tr>
<tr>
<td><code>sub(.b)</code></td>
<td><code>inc(.b)</code></td>
<td></td>
<td><code>ret</code></td>
</tr>
<tr>
<td><code>subc(.b)</code></td>
<td><code>incd(.b)</code></td>
<td></td>
<td><code>reti</code></td>
</tr>
<tr>
<td><code>xor(.b)</code></td>
<td><code>dec(.b)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>decd(.b)</code></td>
<td><code>adc(.b)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>sbc(.b)</code></td>
<td><code>clr(.b)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>clr(.b)</code></td>
<td><code>dadc(.b)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>rla(.b)</code></td>
<td><code>rlc(.b)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>tst(.b)</code></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Unified Memory Map

- Absolutely no paging
- Supports code agility
- ISP Flash
  - Self programming
  - JTAG
  - Bootloader

```c
// Flash In System Programming
FCTL3 = FWKEY;  // Unlock
FCTL1 = FWKEY | WRT;  // Enable
*(unsigned int *)0xFC00 = 0x1234;
```
Programming 'F2131 8KB Flash?

- \( f_{FTG} \) = ________?
- \( t_{WORD} \) = ______?
- Program word or byte = ________________?

Flash Memory

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>TEST CONDITIONS</th>
<th>VCC</th>
<th>MIN</th>
<th>NOM</th>
<th>MAX</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_{CC} ) (PGM/ERASE)</td>
<td>Program and Erase supply voltage</td>
<td></td>
<td>2.2</td>
<td>3.6</td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>( f_{FTG} )</td>
<td>Flash Timing Generator frequency</td>
<td>257</td>
<td>476</td>
<td></td>
<td>kHz</td>
<td></td>
</tr>
<tr>
<td>( I_{PGM} )</td>
<td>Supply current from ( V_{CC} ) during program</td>
<td>2.7 V/3.6 V</td>
<td>3</td>
<td>5</td>
<td></td>
<td>mA</td>
</tr>
<tr>
<td>( I_{ERASE} )</td>
<td>Supply current from ( V_{CC} ) during erase</td>
<td>2.7 V/3.6 V</td>
<td>3</td>
<td>7</td>
<td></td>
<td>mA</td>
</tr>
<tr>
<td>( t_{CP} )</td>
<td>Cumulative program time</td>
<td>see Note 1</td>
<td></td>
<td></td>
<td>4</td>
<td>ms</td>
</tr>
<tr>
<td>( t_{CM_Erase} )</td>
<td>Cumulative mass erase time</td>
<td>2.7 V/3.6 V</td>
<td></td>
<td></td>
<td>20</td>
<td>ms</td>
</tr>
<tr>
<td>( t_{Retention} )</td>
<td>Program/Erase endurance</td>
<td>( T_J = 25^\circ C )</td>
<td></td>
<td></td>
<td>( 10^4 ) , ( 10^5 )</td>
<td>cycles</td>
</tr>
<tr>
<td>( t_{Word} )</td>
<td>Word or byte program time</td>
<td></td>
<td></td>
<td></td>
<td>100</td>
<td>years</td>
</tr>
<tr>
<td>( t_{Block, \ 0} )</td>
<td>Block program time for 1st byte or word</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{Block, \ 1-63} )</td>
<td>Block program time for each additional byte or word</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{Block, \ End} )</td>
<td>Block program end-sequence wait time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t_{Mass \ Erase} )</td>
<td>Mass erase time</td>
<td></td>
<td></td>
<td></td>
<td>10593</td>
<td></td>
</tr>
<tr>
<td>( t_{Seg \ Erase} )</td>
<td>Segment erase time</td>
<td></td>
<td></td>
<td></td>
<td>4819</td>
<td></td>
</tr>
</tbody>
</table>

NOTES:

1. The cumulative program time must not be exceeded when writing to a 64-byte flash block. This parameter applies to all programming methods: individual word/byte write and block write modes.
2. These values are hardwired into the Flash Controller’s state machine (\( f_{FTG} = 1 / f_{FTG} \)).
Reset Conditions

- RST/NMI configured in the reset mode
- I/O pins are switched to input
- Watchdog timer powers up as active watchdog
- Other peripheral modules are disabled
- Status register (SR) is reset
- Program counter (PC) is loaded with (0FFFEh)

Always refer to Family users guide
ATC Board

- Default jumper settings

- MSP430FG4619
  - ‘4619 JTAG

- MSP430F2013
  - ‘2013 JTAG
IAR Kickstart Tour

• Instructor-led tour of Kickstart
Getting Started Lab1: Blinky

- Blink the LED with assembly code
Lab1: Flash The LED First Program

```c
#include "msp430x20x3.h"

ORG 0F800h                  ; Program start
RESET     mov.w #280h,SP                ; Stack
           mov.w #WDTPW+WDTHOLD,&WDTCTL  ; Stop watchdog
           bis.b #01h,&P1DIR

Mainloop xor.b #01h,&P1OUT
      dec.w R15
      jnz Delay
      jmp Mainloop

Delay      ORG 0FFFEh                  ; RESET vector
           DW     RESET
           END
```
Lab1: Step-by-Step

- Connect the FET interface to the PC USB port
- Connect the JTAG cable to the ‘2013 JTAG port on the board
- Connect the BATT jumper on your board
- Launch IAR
- Create new workspace
  - File → New → Workspace
- Create new project
  - Project → Create new project → Click “OK” on dialogue pop-up
  - Name the project (Lab1) and click “Save”
Lab1: Step-by-Step (cont.)

• Configure the project
  ▪ Project → Options
    – Select MSP430F2013 from “Device” drop down menu (DDM)
    – Click “Assembler Only” project
    – Highlight “Debugger” in the Category list
    – Select “FET Debugger” from the “Driver” DDM
    – Highlight “FET Debugger” in the Category list
    – Select the “TI USB” in the “Connection” section
    – Click OK

• Create the source file
  ▪ File → New → File
  ▪ Type source from slide
  ▪ Click save, name it, and save it
Lab1: Step-by-Step (cont.)

• Add source file to the project
  ▪ Project → Add Files
  ▪ Select “Assembler Files” from “Files Of Type” DDM
  ▪ Select your file and click Open

• Click the “Debug” button in IAR:

• Name and save workspace when prompted

• Click “OK” about the Stack Plug-in

• Click “Go” in IAR:

• Your done! Click “Stop Debugging” button:

• Pull the power jumper
Lab1: For Future Reference

2.2.2 Creating a Project from Scratch

The following section presents step-by-step instructions to create an assembler or C project from scratch, and to download and run the application on the MSP430. Refer to Project Settings above. Also, the MSP430 IAR Embedded Workbench IDE User Guide presents a more comprehensive overview of the process.

• Previous instructions summarized section 2.2.2 of FET User’s Guide (navigate from START menu)
Agenda

• Intro to MSP430 Architecture
• Intro to Tools and I/O
• Intro to Low-Power with the MSP430
Embedded Emulation

• Debug real time in system
  ▪ No application resources used
  ▪ Full speed
  ▪ Breakpoint
  ▪ Single step
  ▪ Complex trigger
  ▪ Trace

• Security Fuse
MSP-FET430

USB FET Interface:

JTAG Interface Target Board

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Product Family</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSP-FET430U28</td>
<td>MSP430x11x1A, MSP430x12x/x1xx2</td>
<td>$149.00</td>
</tr>
<tr>
<td>MSP-FET430U64</td>
<td>MSP430x13x/x14x/x15x/x16x</td>
<td>$149.00</td>
</tr>
<tr>
<td>MSP-FET430U80</td>
<td>80-pin MSP430x43x/MSP430x44x</td>
<td>$149.00</td>
</tr>
<tr>
<td>MSP-FET430U100</td>
<td>MSP430FG43x, MSP430x43x</td>
<td>$149.00</td>
</tr>
</tbody>
</table>

Interface only without target board:

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Product Family</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSP-FET430UIF</td>
<td>MSP430</td>
<td>$99.00</td>
</tr>
</tbody>
</table>
MSP-FET430

Parallel FET Interface:

JTAG Interface

Target Board

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Product Family</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSP-FET430P120</td>
<td>MSP430x11x1A, MSP430x12x/x1xx2</td>
<td>$99.00</td>
</tr>
<tr>
<td>MSP-FET430P140</td>
<td>MSP430x13x/x14x/x15x/x16x</td>
<td>$99.00</td>
</tr>
<tr>
<td>MSP-FET430P410</td>
<td>MSP430x41x, MSP430FE42x, MSP430FW42x</td>
<td>$99.00</td>
</tr>
<tr>
<td>MSP-FET430P430</td>
<td>MSP430FG43x, MSP430x43x</td>
<td>$99.00</td>
</tr>
<tr>
<td>MSP-FET430P440</td>
<td>MSP430F43x/44x</td>
<td>$99.00</td>
</tr>
</tbody>
</table>

Interface only without target board:

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Product Family</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSP-FET430PIF</td>
<td>MSP430</td>
<td>$49.00</td>
</tr>
</tbody>
</table>
eZ430-F2013

Spy Bi-Wire Interface

All target pins accessible

USB Powered

LED

Emulator

Removable Target Board

MSP430F2013
IAR Kickstart IDE

- MSP-FET430 IDE
  - Assembler/linker
  - 4kB IAR compiler
  - Parallel or USB Interface

© 2006 Texas Instruments Inc, Slide 34
Which IDE Do I Use?

- IAR Embedded Workbench
- TI Code Composer Essentials
- Rowley, Quadravox, Image Craft, GCC, Others

Check Third-Party website for complete list

- Most have 30-day trials

Check the Yahoo! user group for recommendations
www.ti.com/msp430

• User’s Guide
• Datasheets
• 100+ Application reports
• 500+ Code examples
• Complete 3rd party listing
• Errata
Code Examples!

- Reduce development time
- Over 1000 free examples
- Provided in C / assembler
- Use standalone
- Use as a template for your next project
Why Use Standard Definitions?

```
WDTCTL = 0x5A80;
WDTCTL = 0xA580;
WDTCTL = 0xA540;  // Hold watchdog
WDTCTL = WDTPW + WDTHOLD;  // Hold watchdog
```

• Which code line holds the watchdog?
Getting Started Lab2: I/O Overview

• Configure Port1 and Port2 of the MSP430FG4619
  ▪ P1.0 as input with interrupt enabled
  ▪ P1.0 interrupt on H-L transition
  ▪ P2.1 as output to turn on LED

• Inside of P1ISR
  ▪ Clear pending interrupt flag
# Port GPIO

<table>
<thead>
<tr>
<th>Input Register PxIN</th>
<th>Output Register PxOUT</th>
<th>Direction Register PxDIR</th>
<th>Function Select PxSEL</th>
<th>Interrupt Edge PxIES</th>
<th>Interrupt Enable PxIE</th>
<th>Interrupt Flags PxIFG</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEST 1</td>
<td>20</td>
<td>P1.7/TA2/TDO/TDI</td>
<td>010h</td>
<td>&amp;P1DIR</td>
<td>&amp;P1SEL</td>
<td></td>
</tr>
<tr>
<td>VCC 2</td>
<td>19</td>
<td>P1.6/TA1/TDI/TCLK</td>
<td>010h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2.5/CA5 3</td>
<td>18</td>
<td>P1.5/TA0/TMS</td>
<td>010h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VSS 4</td>
<td>17</td>
<td>P1.4/SMCLK/TCK</td>
<td>010h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XOUT/P2.7/CA7 5</td>
<td>16</td>
<td>P1.3/TA2</td>
<td>010h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XIN/P2.6/CA6 6</td>
<td>15</td>
<td>P1.2/TA1</td>
<td>010h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RST/NMI 7</td>
<td>14</td>
<td>P1.1/TA0</td>
<td>010h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2.0/ACLK/CA2 8</td>
<td>13</td>
<td>P1.0/TACLK</td>
<td>010h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2.1/INCLK/CA3 9</td>
<td>12</td>
<td>P2.4/TA2/CA1</td>
<td>010h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2.2/CAOUT/TA0/CA4</td>
<td>11</td>
<td>P2.3/TA1/CA0</td>
<td>010h</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** P1 and P2 only

- `bis.b #010h, &P1DIR`
- `bis.b #010h, &P1SEL`
- `bis.b #001h, &P1DIR`
- `bis.b #001h, &P1OUT`

© 2006 Texas Instruments Inc, Slide 40
Lab2: Step-by-Step

• Move JTAG cable to ‘4619 and connect BATT jumper
• Launch IAR
• Open previous workspace when prompted
• Create new project as before
  ▪ Project → Create new project → Click “OK” on dialogue pop-up
  ▪ Name the project (Lab2) and click “Save”

• Configure the project
  ▪ Project → Options
    – Select MSP430FG4619 from “Device” drop down menu (DDM)
    – Highlight “Debugger” in the Category list
    – Select “FET Debugger” from the “Driver” DDM
    – Highlight “FET Debugger” in the Category list
    – Select the “TI USB” in the “Connection” section
    – Click OK
Lab2: Step-by-Step (cont.)

• Add source file to the project
  ▪ Project → Add Files
  ▪ Select “Getting_Started_Lab2.c” and click Open

• Complete the code

• Click the “Debug” button in IAR:

• Click “Go” in IAR:

• Test and debug your code

• When done, click “Stop Debugging” button:

• Pull the power jumper
void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;  // Stop WDT
    FLL_CTL0 |= XCAP14PF;  // Configure load caps
    P2DIR = ____;  // Set P2.1 to output direction
    P1IES = ____;  // H-L transition
    P1IE = ____;  // Enable interrupt
    _EINT();  // Enable interrupts
    while (1);
}

// P1 interrupt service routine
#pragma vector=PORT1_VECTOR
__interrupt void P1ISR (void)
{
    ..........  
    P1IFG &= ~____;  // Clear P1IFG
}
void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;  // Stop WDT
    FLL_CTL0 |= XCAP14PF;      // Configure load caps
    P2DIR = BIT1;              // Set P2.1 to output direction
    P1IES = BIT0;              // H-L transition
    P1IE = BIT0;               // Enable interrupt
    _EINT();                   // Enable interrupts
    while (1);
}

// P1 interrupt service routine
#pragma vector=PORT1_VECTOR
__interrupt void P1ISR (void)
{
        P1IFG &= ~BIT0;         // Clear P1IFG
}
Agenda

• Intro to MSP430 Architecture
• Intro to Tools and I/O
• Intro to Low-Power with the MSP430
Ultra-low Power Activity Profile

250uA

Active

Active

Standby

1uA

Always-on

On demand

MSP430

ACLK

low-power peripherals

32768

DCO

MCLK

CPU and peripherals

© 2006 Texas Instruments Inc, Slide 46
Performance on Demand

Interrupt

DCO

Immediate-stable clock start for reaction to events

Δ: 400mV
@: 3.04 V
Δ: 204ns
@: 2.00ns
Ch2 Freq 15.92MHz
Ultra-low Power Clock Control

- **LPM0**
  - CPU Off
  - DCO on
  - ACLK on
  - 35uA

- **LPM3**
  - RTC function
  - LCD driver
  - RAM/SFR retained

- **Active**
  - DCO on
  - ACLK on
  - 250uA

- **Stand-by**
  - DCO off
  - ACLK on
  - 0.8uA

- **Off**
  - All Clocks Off
  - 0.1uA

- **LPM4**
  - RAM/SFR retained

<6us transitions between states.
MSP430x11x/12x Basic Clock

R2/SR

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

LFXT1 Oscillator

OSCOFF

12pF

12pF

LF

XT

DCO

Digitally Controlled Oscillator

ZNGIECPU

OFF

OSC

OFF

CPU

OFF

GIE

N

Z

C

VCC

MCLK

SMCLK

 ACLK

CPUOFF

SCG1

SCG0
‘1xx Basic Clock XTAL Options

- Most MSP430 applications use a 32768 XTAL
'1xx Basic Clock DCO Control

DCOCLK

10MHz
1MHz
100kHz

RSELx

-0.33%/C

MODx

1

n

n+1

Average DCO

XT2OFF  XTS  DIVA1  DIVA0  XT5  RSEL2  RSEL1  RSEL0  DCO2  DCO1  DC00  MOD4  MOD3  MOD2  MOD1  MOD0

Technology for Innovators™
Does the DCO Have Jitter?

- What are the benefits of mixing two frequencies?
'1xx DCO Calibration

Periodic loop can adjust DCO

If Rosc = 100k then DCOCLK ~ 2MHz
'4xx FLL

- OSCfault fail-safe for LFXT1, DCO and XT2
## Low Power Mode Configuration

<table>
<thead>
<tr>
<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>R2/SR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Active Mode</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>~ 250uA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LPM0</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td>~ 35uA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LPM3</strong></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td>~ 0.8uA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LPM4</strong></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>~ 0.1uA</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
bis.w  #CPUOFF,SR ; LPM0
```
## Interrupt Vectors – ‘F11x1

<table>
<thead>
<tr>
<th>SOURCE initialization</th>
<th>FLAG</th>
<th>INTERRUPT</th>
<th>ADDRESS</th>
<th>PRIORITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power-up ext. Reset Watchdog</td>
<td>WDTIFG</td>
<td>Reset</td>
<td>0FFFEh</td>
<td>15, highest</td>
</tr>
<tr>
<td>NMI Osc. Fault Flash violation</td>
<td>NMIIFG</td>
<td>(non)-maskable</td>
<td>0FFFAh</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>OFIFG</td>
<td>(non)-maskable</td>
<td>0FFF8h</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>ACCVIFG</td>
<td>(non)-maskable</td>
<td>0FFF6h</td>
<td>11</td>
</tr>
<tr>
<td>Comparator_A</td>
<td>CAIFG</td>
<td>maskable</td>
<td>0FFF4h</td>
<td>10</td>
</tr>
<tr>
<td>Watchdog timer</td>
<td>WDTIFG</td>
<td>maskable</td>
<td>0FFF2h</td>
<td>9</td>
</tr>
<tr>
<td>Timer_A</td>
<td>CCIFG0</td>
<td>maskable</td>
<td>0FFF0h</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>CCIFGx</td>
<td>maskable</td>
<td>0FFEEh</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0FFECh</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0FFE8h</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0FFDh</td>
<td>4</td>
</tr>
<tr>
<td>I/O Port P2</td>
<td>P2IFGx</td>
<td>maskable</td>
<td>0FFE6h</td>
<td>3</td>
</tr>
<tr>
<td>I/O Port P1</td>
<td>P1IFGx</td>
<td>maskable</td>
<td>0FFE4h</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0FFE2h</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0FFE0h</td>
<td>0, lowest</td>
</tr>
</tbody>
</table>

### FLASH
(x) 512B Segments

- (2) 128B
- Boot Loader
- RAM
- 16-bit Peripherals
- 8-bit Peripherals

© 2006 Texas Instruments Inc, Slide 56
Interrupt Processing

Prior to ISR

ISR hardware - automatically
- PC pushed
- SR pushed
- Interrupt vector moved to PC
- GIE, CPUOFF, OSCOFF and SCG1 cleared
- IFG flag cleared on single source flags

reti - automatically
- SR popped - original
- PC popped
Low Power Modes In Assembler

RESET
- org 0F000h
- 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
Low Power Modes In C

```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);
}
```

Technology for Innovators™
Interrupts Control Program Flow

9600 baud

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

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

100% CPU Load

0.1% CPU Load
Software Functions >> Peripherals

MCU
P1.2

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

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

100% CPU Load

Zero CPU Load

© 2006 Texas Instruments Inc, Slide 61
Power Manage External Devices

- OPA with shutdown can be 20x lower total power

0.01uA = Shutdown
20uA = Active

0.06uA = Average

1uA = Quiescent
1uA = Active

1uA = Average

© 2006 Texas Instruments Inc, Slide 62
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
How To Terminate Unused Pins?

• Unused port pins Px.0 – Px.7?
  ▪ Set as output direction avoids floating gate current.

• XT2IN, XT2OUT?

• Please see last page of chapter 2 in user’s guide.
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
• Every unnecessary instruction executed is a portion of the battery wasted that will never return.
Getting Started Lab3: Low-Power

- Lab2 has been converted to use LPM3 instead of the while(1) wait loop
- Open IAR and create a new project as before
- Add the file “Getting_Started_Lab3.c” to the project
- Download the code as before
- Disconnect the JTAG interface
- Measure the current through the PWR1 jumper
while(1)
{
  _BIS_SR(LPM3_bits);    // Enter LPM3
  if ((P1IN & 0x01) == 0)
    P2OUT ^= 0x02;        // Toggle P2.1 using exclusive-OR
}

// P1 interrupt service routine
#pragma vector=PORT1_VECTOR
__interrupt void P1ISR (void)
{
  unsigned volatile int i;
  for (i=10000; i>0; i--);  // Debounce delay
  P1IFG &= ~BIT0;          // Clear P1IFG
  _BIC_SR_IRQ(LPM3_bits);  // Clear LPM3 bits from 0(SR)
Lab3: Low-Power Questions

• What is the current through the PWR1 jumper?
• 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?
Lab3: Low-Power Answers

• What is the current through the PWR1 jumper?
  ▪ About 1.4 uA

• Why were the I/Os configured as they were?
  ▪ Unused I/O must be configured as outputs, otherwise, floating gate current will be seen. The outputs were then set to values so as not to contend with other on-board circuitry

• Why was LPM3 used?
  ▪ No clocks are needed. LPM3 leaves on the 32768Hz running and shuts down all other clocks.

• Look in the header file to see how LPM3_bits is defined
  ▪ SCG1+SCG0+CPUOFF

• What further low-power improvements could be made?
  ▪ LPM4 could be used. A timer could be employed for the debounce
Lab3: Going Further

- Convert the code to use LPM4 and re-measure the current. What is it now?
  - Should be about .3 uA
IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI’s standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, including any liability arising out of their use. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or “enhanced plastic.” Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer’s risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

<table>
<thead>
<tr>
<th>Products</th>
<th>Applications</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amplifiers</td>
<td>Audio</td>
<td><a href="amplifier.ti.com">amplifier.ti.com</a></td>
</tr>
<tr>
<td>Data Converters</td>
<td>Automotive</td>
<td><a href="www.ti.com/automotive">www.ti.com/automotive</a></td>
</tr>
<tr>
<td>DSP</td>
<td>Broadband</td>
<td><a href="www.ti.com/broadband">www.ti.com/broadband</a></td>
</tr>
<tr>
<td>Interface</td>
<td>Digital Control</td>
<td><a href="www.ti.com/digitalcontrol">www.ti.com/digitalcontrol</a></td>
</tr>
<tr>
<td>Logic</td>
<td>Military</td>
<td><a href="www.ti.com/military">www.ti.com/military</a></td>
</tr>
<tr>
<td>RFID</td>
<td>Telephony</td>
<td><a href="www.ti.com/telephony">www.ti.com/telephony</a></td>
</tr>
<tr>
<td>Low Power</td>
<td>Video &amp; Imaging</td>
<td><a href="www.ti.com/video">www.ti.com/video</a></td>
</tr>
<tr>
<td>Wireless</td>
<td></td>
<td><a href="www.ti.com_wireless">www.ti.com_wireless</a></td>
</tr>
</tbody>
</table>

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2007, Texas Instruments Incorporated