THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

CPE 323: Power, MSP430 Low Power Modes

Aleksandar Milenkovic

Electrical and Computer Engineering
The University of Alabama in Huntsville

milenka@ece.uah.edu

http://www.ece.uah.edu/~milenka

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Outline

- Introduction
- Background: Power in CMOS Integrated Circuits
- Power in Embedded Systems
- MSP430 Operating Modes (Active, LPMs)
- Demos

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Power as a Design Constraint

- Power: work done per unit time (VQ/t = VI)
- Why worry about power?
  - Battery life in portable and mobile platforms
  - Power consumption in desktops, server farms
    - Cooling costs, packaging costs, reliability, timing
    - Power density: 30 W/cm2 in Alpha 21364 (3x of typical hot plate)
  - Environment?
    - IT consumes 10% of energy in the US

Power becomes a first class architectural design constraint

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Background: Where does power go in CMOS?

Dynamic power consumption

Power due to short-circuit current during transition

Power due to leakage current

$$P = ACV^2f + \tau AVI_{short}f + VI_{leak}$$

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Dynamic Power Consumption

C – Total capacitance seen by the gate's outputs Function of wire lengths, transistor sizes, ...

V – Supply voltage Trend: has been dropping with each successive fab

$$ACV^2f$$

A - Activity of gates How often on average do wires switch?

f – clock frequency Trend: increasing ...

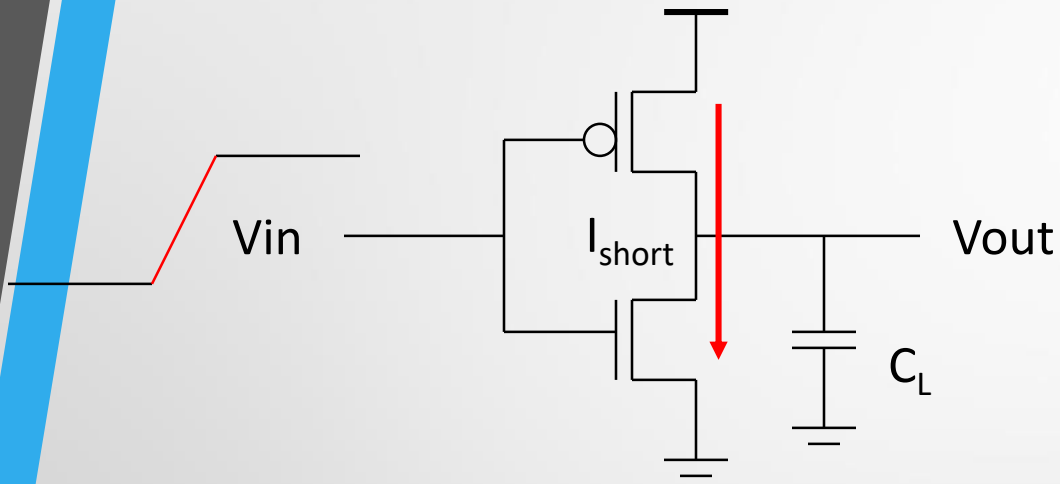
Reducing Dynamic Power

1) **Reducing V has quadratic effect; Limits?**
2) **Lower C - shrink structures, shorten wires**
3) **Reduce switching activity - Turn off unused parts or use design techniques to minimize number of transitions**

LaCASA

Short-circuit Power Consumption

$$\tau A V I_{short} f$$



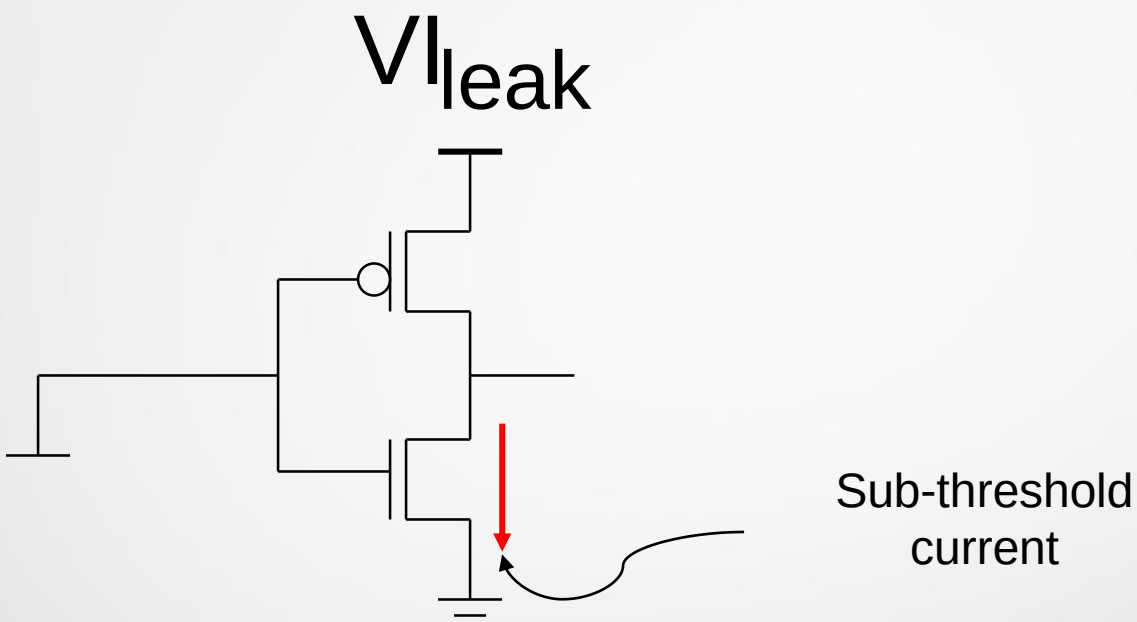
Finite slope of the input signal causes a direct current path between V_{DD} and GND for a short period of time during switching when both the NMOS and PMOS transistors are conducting

Reducing Short-circuit

1) **Lower the supply voltage V**
2) **Slope engineering – match the rise/fall time of the input and output signals**

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Leakage Power



Sub-threshold current grows exponentially with increases in temperature and decreases in Vt

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

CMOS Power Equations

$$P = ACV^2f + \tau AVI_{short}f + VI_{leak}$$

Reduce the supply voltage, V

$$f_{max} \propto \frac{(V - V_t)^2}{V}$$

Reduce threshold V_t

$$I_{leak} \propto \exp(-\frac{qV_t}{kT})$$

LaCASA

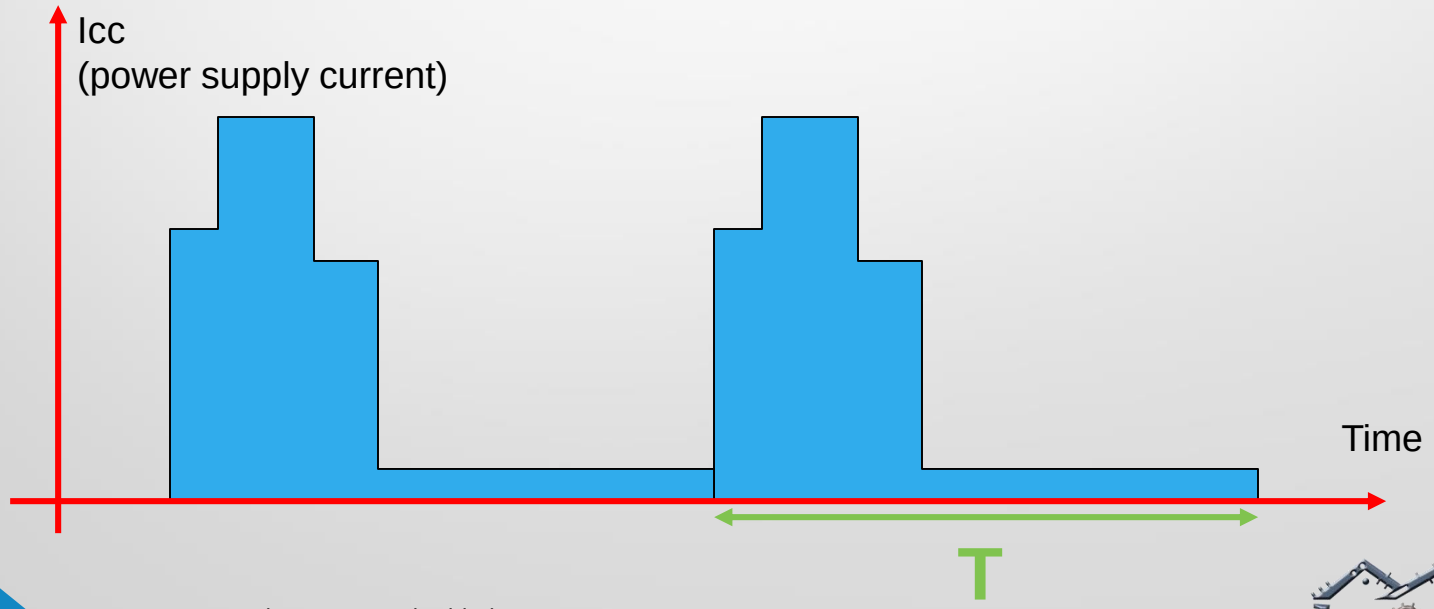
How Can We Reduce Power Consumption?

- Dynamic power consumption
  - Fan-out: charge/discharge of the capacitive load on each gate's output
  - Frequency
- Control activity
  - Reduce power supply voltage
  - Reduce working frequency
  - Turn off unused parts (module enables)
  - Use low power modes
  - Interrupt driven system
- Minimize the number of transitions
  - Instruction formats, coding?

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Average Power Consumption

- Dynamic power supply current
  - Set of modules that are periodically active
  - Typical situation – real time cycle T
  - Iave = $\int$ Icc(t)dt /T
  - Most cases Iave = Σ Ii*ti/T



LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Low-Power Concept: Basic Conditions for Burst Mode

The example of the heat cost allocator shows that the current of the non-activity periode dominates the current consumption.

		Measure		Process data		Real-Time Clock		LCD Display
I_{AVG}	=	$I_{Measure}$	+	$I_{Calculate}$	+	I_{RTC}	+	$I_{Display}$
	=	$I_{ADC} * t_{Measure}/T$	+	$I_{active} * t_{calc}/T$	+	$I_{active} * t_{RTC}/T$	+	$I_{Display}$
	=	3mA *200µs/60s	+	0.5mA * 10ms/60s	+	0.5mA * 0.5ms/60s	+	2.1µA
	=	10nA	+	83nA	+	4nA	+	2.1µA
I_{AVG}	≅							2.1µA

The ***sleep current dominates*** the current consumption!

The currents are related to the sensor and µC system. Additional current consumption of other system parts should be added for the total system current

LaCASA

Battery Life

- Battery Capacity BC – [mAh]
- Battery Life
  - BL = BC / Iavg
- In the previous example, standard 800 mAh batteries will allow battery life of:
  - BL = 750 mAh / 2.1 μA ≈ 44 years !!!
- Conclusion:
  - Power efficient modes
  - Interrupt driven system with processor in idle mode

Power and Related Metrics

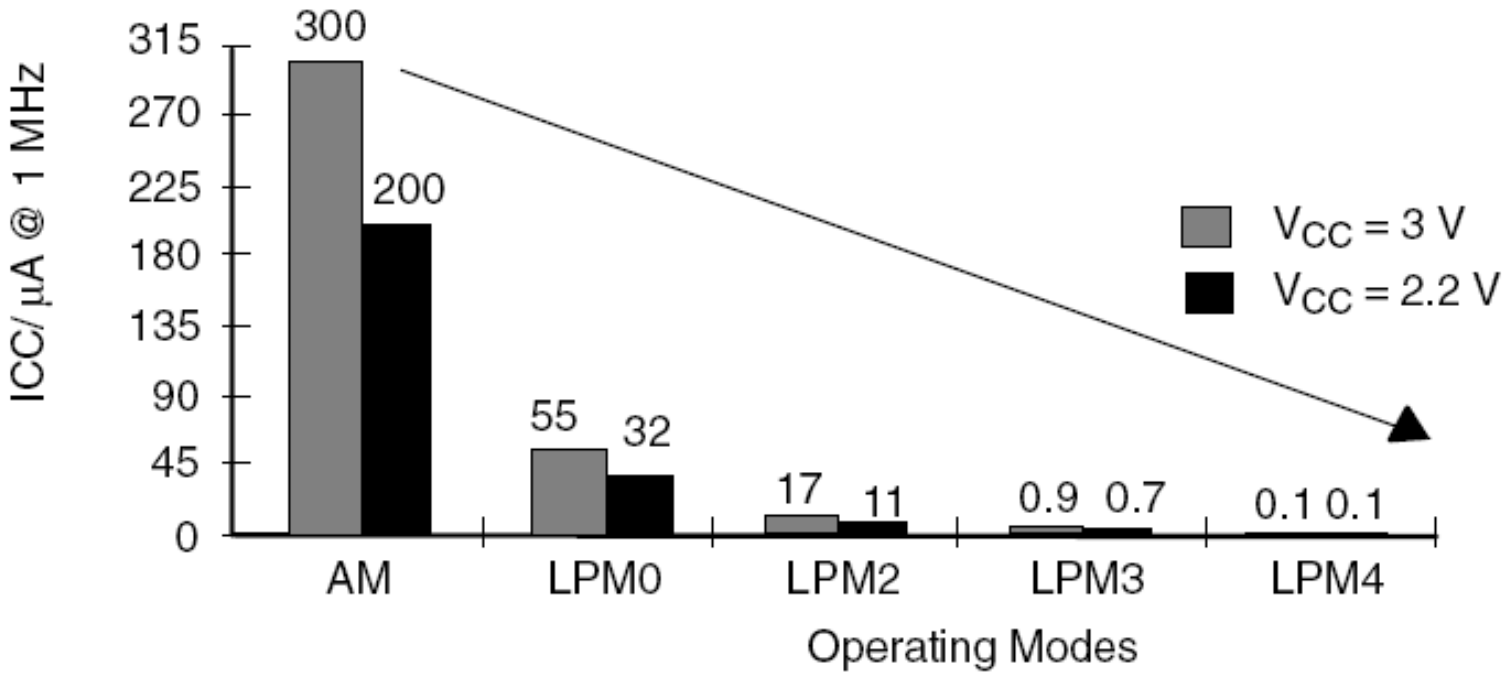
- Peak power
  - Possible damage
- Dynamic power
  - Non-ideal battery characteristics
  - Ground bounce, di/dt noise
- Energy/operation ratio
  - MIPS/W
  - Energy x Delay

Operating Modes in MS430

- Support low- and ultra-low energy consumption
  - Intelligent management of individual modules and clock subsystems
- An interrupt event wakes the system from each of the various operating modes and the RETI instruction returns operation to the mode that was selected before the interrupt event
- Balance requirements
  - The desire for speed and data throughput despite conflicting needs for ultra-low power
  - Minimization of individual current consumption
  - Limitation of the activity state to the minimum required by the use of low power modes

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

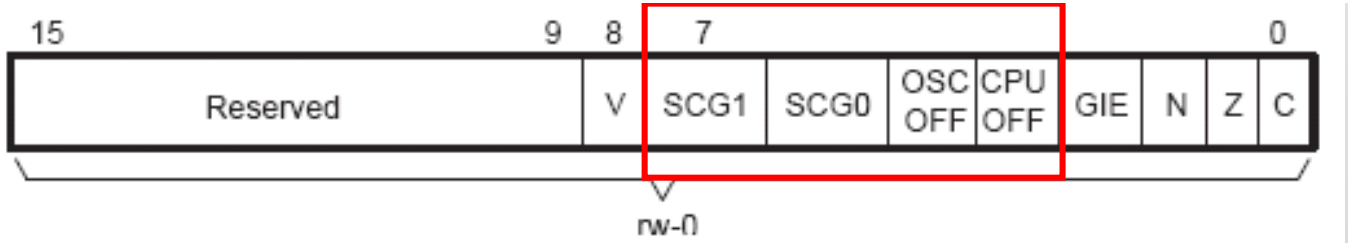
Current (F, Mode, V)



LaCASA

Operating Modes in MSP430 (cont'd)

SCG1	SCG0	OSCOFF	CPUOFF	Mode	CPU and Clocks Status
0	0	0	0	Active	CPU is active, all enabled clocks are active
0	0	0	1	LPM0	CPU, MCLK are disabled (41x/42x peripheral MCLK remains on) SMCLK , ACLK are active
0	1	0	1	LPM1	CPU, MCLK, DCO oscillator are disabled (41x/42x peripheral MCLK remains on) DC generator is disabled if the DCO is not used for MCLK or SMCLK in active mode SMCLK , ACLK are active
1	0	0	1	LPM2	CPU, MCLK, SMCLK, DCO oscillator are disabled DC generator remains enabled ACLK is active
1	1	0	1	LPM3	CPU, MCLK, SMCLK, DCO oscillator are disabled DC generator disabled ACLK is active
1	1	1	1	LPM4	CPU and all clocks disabled



Low Power Mode Control

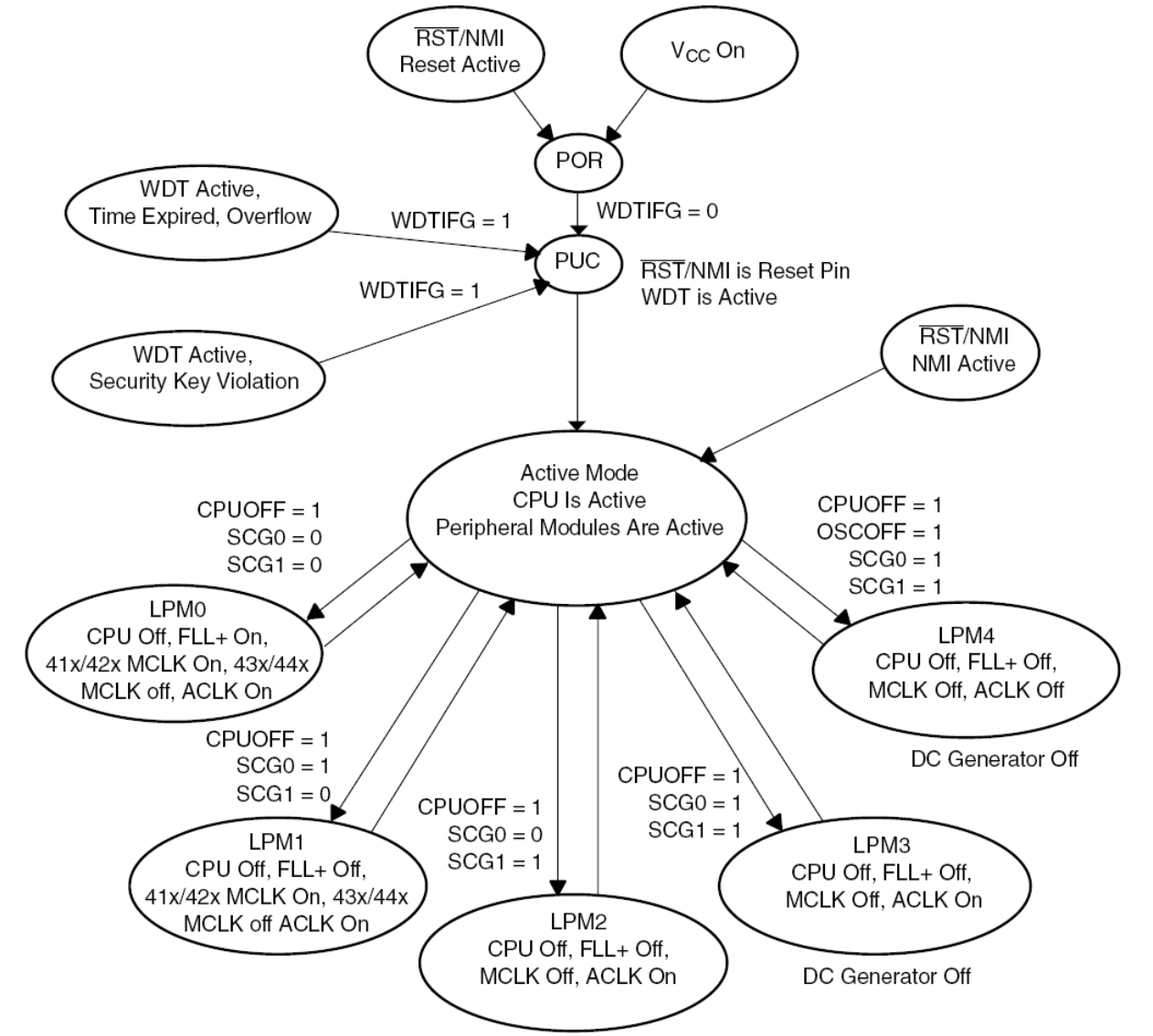
- CPUOff: CPU Off (bit 4 in SR)
  - When set, turns off the CPU
- OscOff: Oscillator Off (bit 5 in SR)
  - When set, turns off the LFXT1 crystal oscillator when LFXT1CLK is not used for MCLK and SMCLK
- SCG0: System Clock Generator 0 (bit 6 in SR)
  - When set, turns off the DCO DC generator if DCOCLK is not used for MCLK and SMCLK
- SCG1: System Clock Generator 1 (bit 7 in SR)
  - When set, turns off the SMCLK

Low Power Mode Control (cont'd)

- Control bits are in SR
- => Present state of the operating condition is saved onto the stack during an interrupt service request
- As long as the stored status register information is not altered, the processor continues (after RETI) with the same operating mode as before the interrupt event

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Operating Modes in MSP430 with FLL+



Operating Modes (AM, LPM0-LMP2)

- Active mode AM; SCG1=0, SCG0=0, OscOff=0, CPUOff=0: CPU clocks are active
- Low power mode 0 (LPM0); SCG1=0, SCG0=0, OscOff=0, CPUOff=1:
  - CPU is disabled/ MCLK is disabled
  - SMCLK and ACLK remain active
- Low power mode 1 (LPM1); SCG1=0, SCG0=1, OscOff=0, CPUOff=1:
  - CPU is disabled/MCLK is disabled
  - DCO's dc generator is disabled if the DCO is not used for MCLK or SMCLK when in active mode. Otherwise, it remains enabled.
  - SMCLK and ACLK remain active
- Low power mode 2 (LPM2); SCG1=1, SCG0=0, OscOff=0, CPUOff=1:
  - CPU is disabled/ MCLK is disabled
  - SMCLK is disabled
  - DCO oscillator automatically disabled because it is not needed for MCLK or SMCLK; DCO's dc-generator remains enabled
  - ACLK remains active

Operating Modes (LPM3-LPM4)

- Low power mode 3 (LPM3); SCG1=1, SCG0=1, OscOff=0, CPUOff=1:
  - CPU is disabled/ MCLK is disabled
  - SMCLK is disabled
  - DCO oscillator is disabled; DCO's dc-generator is disabled
  - ACLK remains active
- Low power mode 4 (LPM4); SCG1=X, SCG0=X, OscOff=1, CPUOff=1:
  - CPU is disabled/ MCLK is disabled
  - ACLK is disabled
  - SMCLK is disabled
  - DCO oscillator is disabled/DCO's dc-generator is disabled
  - Crystal oscillator is stopped

What Happens in a LPM0/LPM1

- Immediately after the bit CPUOff is set the CPU stops operation
  - CPU halts and all internal bus activities stop until an interrupt request or reset occurs
  - System clock generator continues operation, and the clock signals DCO, SMCLK, and ACLK stay active depending on the state of the other three status register bits, SCG0, SCG1, and OscOff
  - Peripherals are enabled or disabled with their individual control register settings, and with the module enable registers in the SFRs
  - All I/O port pins and RAM/registers are unchanged. Wake up is possible through all enabled interrupts

What Happens in a LPM4

- All activities cease; only the RAM contents, I/O ports, and registers are maintained
- Wake up is only possible by enabled external interrupts
- Before activating LPM4, the software should consider the system conditions during the low power mode period. The two most important conditions are environmental (that is, temperature effect on the DCO), and the clocked operation conditions.
- The environment defines whether the value of the frequency integrator should be held or corrected. A correction should be made when ambient conditions are anticipated to change drastically enough to increase or decrease the system frequency while the device is in LPM4

Enter ISR

- Enter LPM: Set corresponding bits
- Process ISR
  - Entered and processed if an enabled interrupt awakens the MSP430
  - The SR and PC are stored on the stack, with the content present at the interrupt event
  - Subsequently, the operation mode control bits OscOff, SCG1, and CPUOff are cleared automatically in the status register

Return from ISR

- Option #1:
  - Return with low-power mode bits set (RETI will restore the original SR)
    - PC points to an instruction that will not be executed, since the restored low power mode stops CPU activity
- Option #2:
  - Return with low-power mode bits reset
    (want to stay in active mode)
  - => ISR must reset the OscOff, CPUOff, SCGO, and SCG1 bits in the original SR on the stack

Operating Modes-Examples

- **The following example describes entering into low-power mode 0.**

```
;===Main program flow with switch to CPUOff Mode==============
BIS #18h,SR ;Enter LPM0 + enable general interrupt GIE
            ;(CPUOff=1, GIE=1). The PC is incremented
            ;during execution of this instruction and
            ;points to the consecutive program step.
......      ;The program continues here if the CPUOff
            ;bit is reset during the interrupt service
            ;routine. Otherwise, the PC retains its
            ;value and the processor returns to LPM0.
```

- **The following example describes clearing low-power mode 0.**

```
;===Interrupt service routine=================================
......                  ;CPU is active while handling interrupts
BIC #10h,0(SP)          ;Clears the CPUOff bit in the SR contents
                        ;that were stored on the stack.
RETI                    ;RETI restores the CPU to the active state
                        ;because the SR values that are stored on
                        ;the stack were manipulated. This occurs
                        ;because the SR is pushed onto the stack
                        ;upon an interrupt, then restored from the
                        ;stack after the RETI instruction.
```

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Operating Modes C Examples

```
#define C                       (0x0001u)
#define Z                       (0x0002u)
#define N                       (0x0004u)
#define V                       (0x0100u)
#define GIE                     (0x0008u)
#define CPUOFF                  (0x0010u)
#define OSCOFF                  (0x0020u)
#define SCG0                    (0x0040u)
#define SCG1                    (0x0080u)

/* Low Power Modes coded with Bits 4-7 in SR */

#ifndef __IAR_SYSTEMS_ICC__ /* Begin #defines for
    assembler */
#define LPM0                    (CPUOFF)
#define LPM1                    (SCG0+CPUOFF)
#define LPM2                    (SCG1+CPUOFF)
#define LPM3                    (SCG1+SCG0+CPUOFF)
#define LPM4                    (SCG1+SCG0+OSCOFF+CPUOFF)
/* End #defines for assembler */

#else /* Begin #defines for C */
#define LPM0_bits               (CPUOFF)
#define LPM1_bits               (SCG0+CPUOFF)
#define LPM2_bits               (SCG1+CPUOFF)
#define LPM3_bits               (SCG1+SCG0+CPUOFF)
#define LPM4_bits               (SCG1+SCG0+OSCOFF+CPUOFF)
```

```
#include "in430.h"

#define LPM0   _BIS_SR(LPM0_bits)     /* Enter Low Power Mode 0 */
#define LPM0_EXIT _BIC_SR_IRQ(LPM0_bits) /* Exit Low Power Mode 0 */
#define LPM1    _BIS_SR(LPM1_bits)     /* Enter Low Power Mode 1 */
#define LPM1_EXIT _BIC_SR_IRQ(LPM1_bits) /* Exit Low Power Mode 1 */
#define LPM2    _BIS_SR(LPM2_bits)     /* Enter Low Power Mode 2 */
#define LPM2_EXIT _BIC_SR_IRQ(LPM2_bits) /* Exit Low Power Mode 2 */
#define LPM3    _BIS_SR(LPM3_bits)     /* Enter Low Power Mode 3 */
#define LPM3_EXIT _BIC_SR_IRQ(LPM3_bits) /* Exit Low Power Mode 3 */
#define LPM4    _BIS_SR(LPM4_bits)     /* Enter Low Power Mode 4 */
#define LPM4_EXIT _BIC_SR_IRQ(LPM4_bits) /* Exit Low Power Mode 4 */
#endif /* End #defines
```


LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

C Examples

```
//*******************************************************************
//  MSP-FET430P140 Demo - WDT Toggle P1.0, Interval ISR, 32kHz ACLK
//
//  Description; Toggle P1.0 using software timed by WDT ISR.
//  Toggle rate is exactly 250ms based on 32kHz ACLK WDT clock source.
//  In this example the WDT is configured to divide 32768 watch-
crystal(2^15)
//  by 2^13 with an ISR triggered @ 4Hz.
//  ACLK= LFXT1= 32768, MCLK= SMCLK= DCO~ 800kHz
//  //*External watch crystal installed on XIN XOUT is required for ACLK*
//
//
//               MSP430F149
//            -----------------
//        /|\|              XIN|-
//         | |                 | 32kHz
//         --|RST          XOUT|-
//           |                 |
//           |             P1.0|-->LED
//
//  M.Buccini
//  Texas Instruments, Inc
//  August 2003
//  Built with IAR Embedded Workbench Version: 1.26B
//  December 2003
//  Updated for IAR Embedded Workbench Version: 2.21B
//*******************************************************************
```

```
#include <msp430x14x.h>


void main(void)

{

 // WDT 250ms, ACLK, interval timer

  WDTCTL = WDT_ADLY_250;

  IE1 |= WDTIE;  // Enable WDT interrupt

  P1DIR |= 0x01; // Set P1.0 to output dir.

   // Enter LPM3 w/interrupt

  _BIS_SR(LPM3_bits + GIE);

}


// Watchdog Timer interrupt service routine

interrupt[WDT_TIMER] void watchdog_timer(void)

{

  P1OUT ^= 0x01;  // Toggle P1.0 using exclusive-OR

}
```


LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

C Examples

```
....
_BIS_SR(LPM0_bits + GIE); // Enter LPM0 w/ interrupt
// program stops here
```

QQ?

Your program is in LPM0 mode and it is woke up by an interrupt. What should be done if you do not want to go back to LPM0 after servicing the interrupt request, but rather you would let the main program re-enter LMP0, based on current conditions?

LaCASA

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

An Example: Wake up in ISR

```
//*******************************************************************
//  MSP430xG46x Demo - FLL+, LPM3 Using Basic Timer ISR, 32kHz ACLK
//
//  Description: System runs normally in LPM3 with basic timer clocked by
//  32kHz ACLK. At a 2 second interval the basic timer ISR will wake the
//  system and flash the LED on P5.1 inside of the Mainloop.
//  ACLK = LFXT1 = 32768Hz, MCLK = SMCLK = default DCO = 32 x ACLK = 104857
//  //* An external watch crystal between XIN & XOUT is required for ACLK *

//
//
//           MSP430xG461x
//         ---------------
//     /|\|              XIN|-
//      | |                 | 32kHz
//      --|RST          XOUT|-
//        |                 |
//        |             P5.1|-->LED
//
//  K. Quiring/ M. Mitchell
//  Texas Instruments Inc.
//  October 2006
//  Built with CCE Version: 3.2.0 and IAR Embedded Workbench Version: 3.41A
//*******************************************************************
#include <msp430xG46x.h>

void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                 // Stop WDT
  FLL_CTL0 |= XCAP14PF;                     // Configure load caps

  IE2 |= BTIE;                              // Enable BT interrupt
  BTCTL = BTDIV+BTIP2+BTIP1+BTIP0;          // 2s Interrupt
  P1DIR = 0xFF;                             // All P1.x outputs
  P1OUT = 0;                                // All P1.x reset
  // repeat for others
  while(1)
  {
    volatile int i;
    _BIS_SR(LPM3_bits + GIE); // Enter LPM3 w/ interrupts
    P5OUT |= 0x02;            // Set P5.1 LED on
    for (i = 5000; i>0; i--); // Delay
    P5OUT &= ~0x02;           // Clear P5.1 LED off
  }
}

// Basic Timer interrupt service routine
#pragma vector=BASICTIMER_VECTOR
__interrupt void basic_timer(void)
{
    _BIC_SR_IRQ(LPM3_bits);    // Clear LPM3 bits from
0(SR)
}
```


LaCASA