

MSP430xxxx Microcontrollers

Low Power Modes

Modes à basse consommation

Power is a priority architectural design constraint

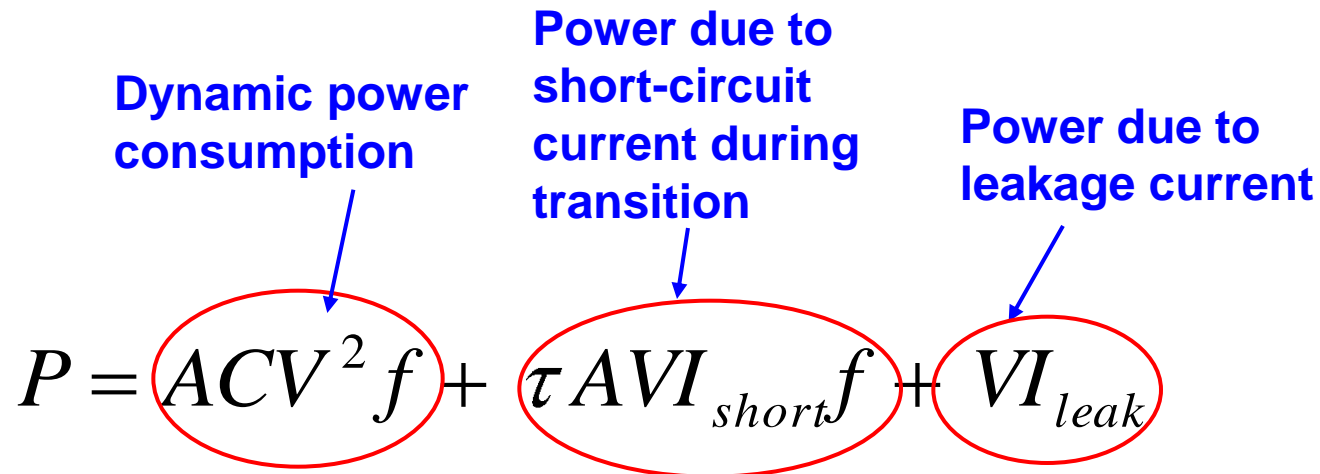
- Why worry about power?
 - Battery life in portable and mobile platforms
 - Power consumption in desktops, server farms
 - Cooling costs, packaging costs, reliability, timing
 - Power density: 30 W/cm² in Alpha 21364 (3x of typical hot plate)
 - Environment?
 - IT consumes 10% of energy in the US

Where does power go in CMOS components ?

Dynamic power consumption

Power due to short-circuit current during transition

Power due to leakage current

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


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 μC generation

$$ACV^2 f$$

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

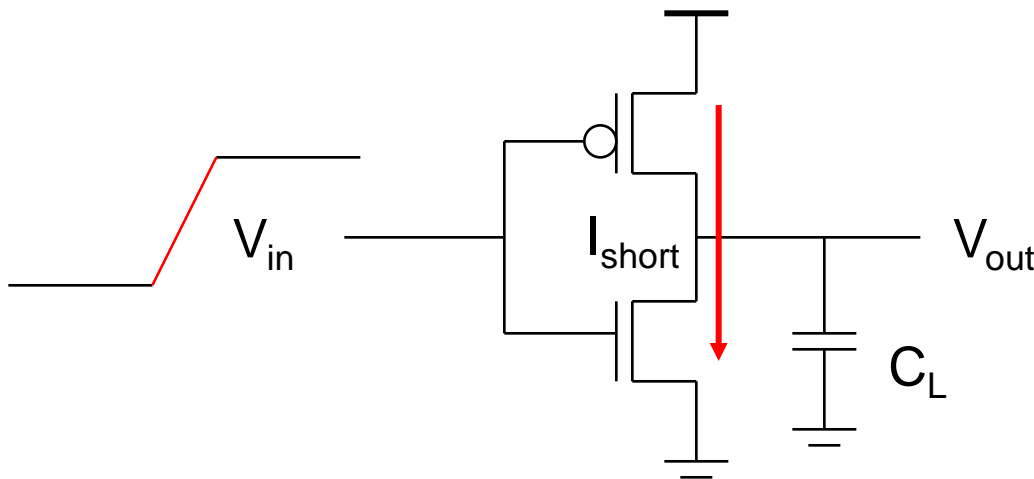
f – clock frequency
Trend: increasing ...

Reducing Dynamic Power

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

Short-circuit Power Consumption

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



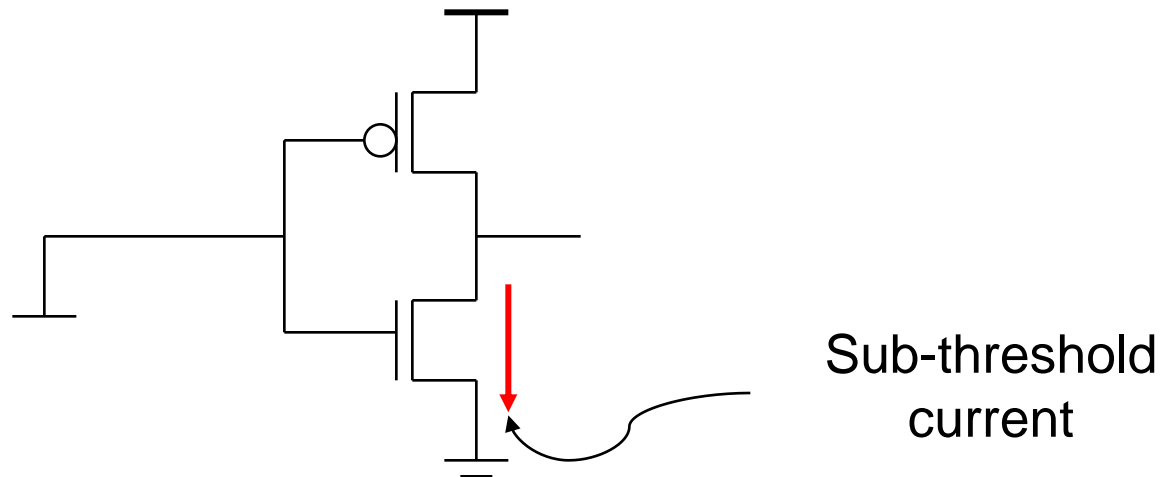
Finite slope of the input signal causes a direct current path between V_{DD} and GND for a short period of time during switching

Reducing Short-circuit

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

Leakage Power

$$VI_{leak}$$



Sub-threshold current grows **exponentially** with increases in **temperature** and **threshold voltage V_t**

CMOS Power Equations

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

Reduce the supply voltage, V

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

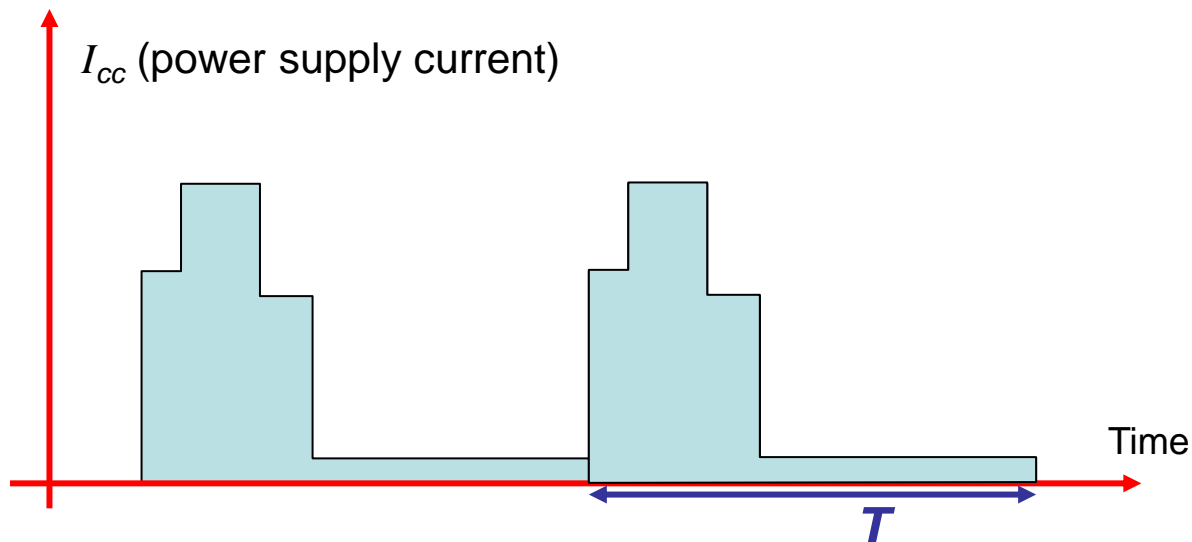
Reduce threshold V_t

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

The average power consumption is an integral ...

- Dynamic power supply current
 - Modules are **periodically** active
 - Typical situation – real time cycle T

$$I_{ave} = \int I_{cc}(t) dt / T$$



- Strategies to take advantage of this:
 - Use of power efficient operation modes
 - Interrupt driven system with processor in idle mode

How can we reduce power consumption?

- Dynamic power consumption
 - lower threshold voltage (affecting charge/discharge of the capacitive load on each gate's output)
 - lower frequency
 - reduce power supply voltage
- Control activity
 - reduce working frequency
 - **use low power modes**
 - interrupt driven system
 - turn off unused parts (module enables)
- Minimize the number of transitions
 - instruction formats, coding?

Aspects allowing to reduce power consumption

1. Logic

- Clock hierachy (up to 30% of power)
- Clock gating (turn off branches that are not used)
- Half frequency clock (both edges)
- Half swing clock (half of V_{CC})
- Asynchronous logic
 - Low power modes
 - Interrupt driven programming

2. Operating System

- Optimize compilation and computation sequence:
“do when necessary”

3. Architecture

- Parallelism (increased area and wiring)
- Power efficient and specialized processing cores
- Speculation (branch prediction)
- Memory systems
 - Memory access (dynamic)
 - Memory banks (turn off unused)
 - Code/data compression

4. Other issues

- Leakage current, thermal runaway

Operating Modes

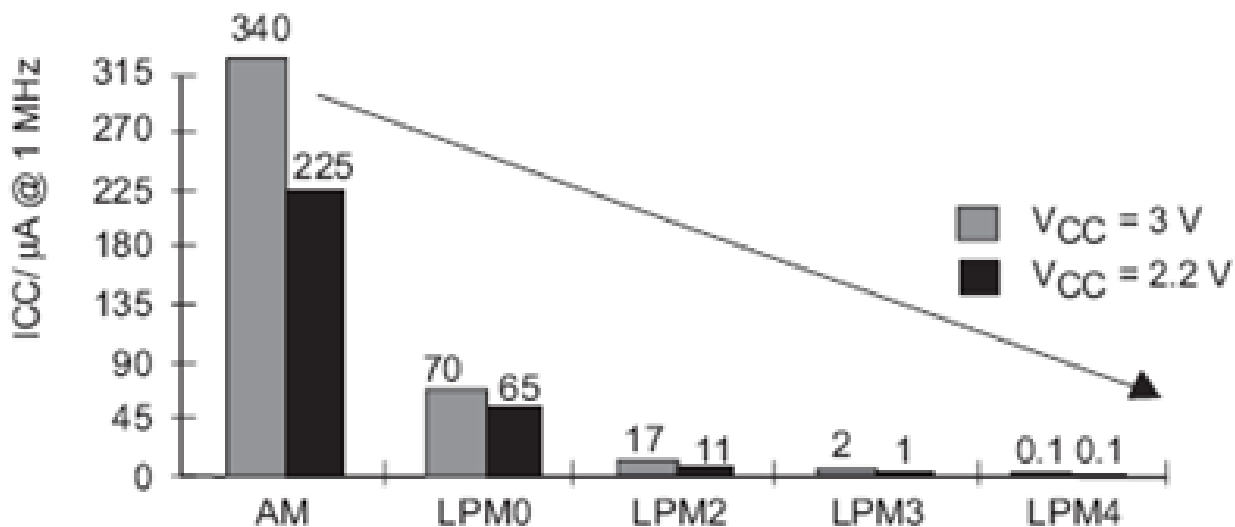
- The MSP430 family was developed for ultralow-power applications and uses different levels of operating modes, which give support to various requirements for ultralow power and ultralow energy consumption.
- This support is combined with an intelligent management of operations during the different module and CPU states.
- An **interrupt event** wakes the system from each of the various operating modes and the RETI – **return ()** instruction returns operation to the mode that was selected before the interrupt event.
- The ultra-low power system design of the MSP430 takes into account three different needs:
 - ❑ 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

Configuration des modes low-power

Les modes low-power 0-4 sont configurés avec les bits CPUOFF, OSCOFF, SCG0, et SCG1 dans le registre de statut (SR).

L'avantage d'inclure les bits de contrôle de mode CPUOFF, OSCOFF, SCG0, and SCG1 dans le SR est que le mode de fonctionnement actuel est sauvé sur le stack lorsqu'une routine de gestion d'interrupt est exécutée.

Fig.: Modes de fonctionnement et consommation associée

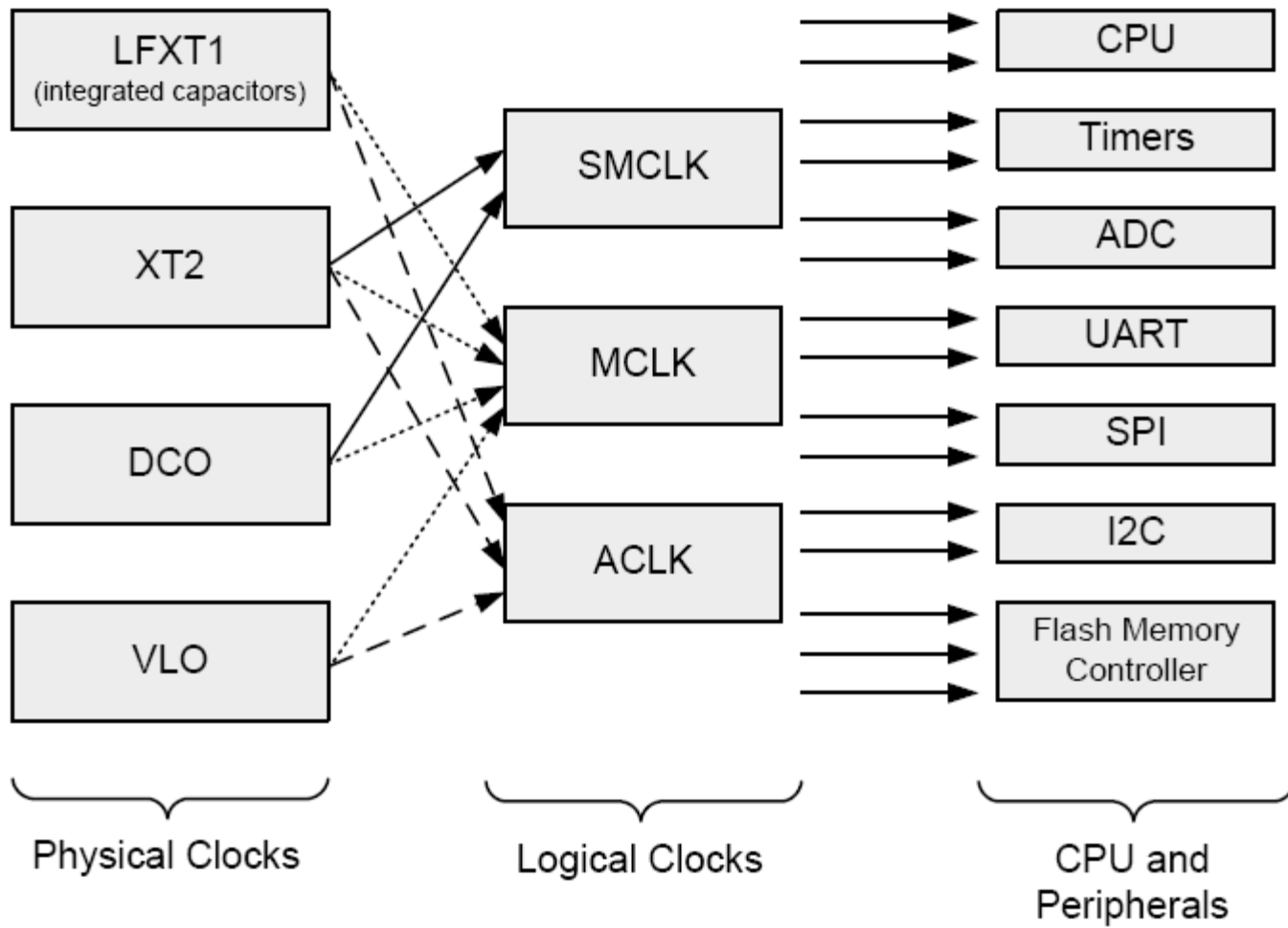


Différents modes et ce qui est actif/inactif

SCG1	SCG0	OSCOFF	CPUOFF	Mode	statut CPU et clock
0	0	0	0	Actif	CPU actif, toutes clocks actives
0	0	0	1	LPM0	CPU, MCLK désactivés SMCLK, ACLK actifs
0	1	0	1	LPM1	CPU, MCLK, DCO osc désactivés SMCLK, ACLK actifs
1	0	0	1	LPM2	CPU, MCLK, SMCLK, DCO osc désactivés DC generator actif ACLK actif
1	1	0	1	LPM3	CPU, MCLK, SMCLK, DCO osc désactivés DC generator désactivé ACLK actif
1	1	1	1	LPM4	CPU et toutes clocks désactivés

(Rappel:)

oscillateurs, horloges, périphériques



Programmation des modes low-power

- Les modes low-power 0-4 sont activés et gérés par les **fonctions qui adressent le SR**, qui sont définies en particulier dans le fichier **intrinsics.h** .

```
__intrinsic void      __bic_SR_register(unsigned short);
__intrinsic void      __bis_SR_register(unsigned short);
__intrinsic unsigned short __get_SR_register(void);
__intrinsic void      __bic_SR_register_on_exit(unsigned short);
__intrinsic void      __bis_SR_register_on_exit(unsigned short);
__intrinsic unsigned short __get_SR_register_on_exit(void);
```

- Plusieurs macros sont définies dans le fichier `intrinsics.h` et sont utiles pour configurer les divers modes dans un programme.

```
// Control bits in the processor status register SR
#define __SR_GIE      (1<<3)
#define __SR_CPU_OFF (1<<4)
#define __SR_OSC_OFF (1<<5)
#define __SR_SCG0    (1<<6)
#define __SR_SCG1    (1<<7)

// Functions for controlling the processor operation modes
#define __low_power_mode_0() (__bis_SR_register( __SR_GIE      \
                                                | __SR_CPU_OFF))

#define __low_power_mode_1() (__bis_SR_register( __SR_GIE      \
                                                | __SR_CPU_OFF  \
                                                | __SR_SCG0))

#define __low_power_mode_2() (__bis_SR_register( __SR_GIE      \
                                                | __SR_CPU_OFF  \
                                                | __SR_SCG1))
```

- (suite ...).

```
#define __low_power_mode_3() \
    (__bis_SR_register(  __SR_GIE \
                        | __SR_CPU_OFF \
                        | __SR_SCG0 \
                        | __SR_SCG1))

#define __low_power_mode_4() \
    (__bis_SR_register(  __SR_GIE \
                        | __SR_CPU_OFF \
                        | __SR_SCG0 \
                        | __SR_SCG1 \
                        | __SR_OSC_OFF))

#define __low_power_mode_off_on_exit() \
    (__bic_SR_register_on_exit(  __SR_CPU_OFF \
                                | __SR_SCG0 \
                                | __SR_SCG1 \
                                | __SR_OSC_OFF))
```

Aussi le fichier **msp430xG46x.h** définit plusieurs autres macros utiles pour activer et désactiver les différents modes :

```
#define CPUOFF          (0x0010)
#define OSCOFF         (0x0020)
#define SCG0           (0x0040)
#define SCG1           (0x0080)

/* Low Power Modes coded with Bits 4-7 in SR */
#define LPM0           (CPUOFF)
#define LPM1           (SCG0+CPUOFF)
#define LPM2           (SCG1+CPUOFF)
#define LPM3           (SCG1+SCG0+CPUOFF)
#define LPM4           (SCG1+SCG0+OSCOFF+CPUOFF)
#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)
#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 */
```

Operating the low-power modes

Being able to access the stack and stack pointer with the instruction set allows the program structures to be individually optimized, as illustrated in the following program flow:

1. Enter interrupt routine

- The interrupt routine is 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.

2. Return from interrupt

- Two different modes are available to return from the interrupt service routine and continue the flow of operation:
 1. **Return with low-power mode bits set.** When returning from the interrupt, the program counter points to the next instruction but the instruction pointed to is not executed, since the restored low power mode stops CPU activity.
 2. **Return with low-power mode bits reset.** When returning from the interrupt, the program continues at the address following the instruction that set the OscOff or CPUOff-bit in the status register.

To use this mode, the interrupt service routine must reset the OscOff, CPUOff, SCGO, and SCG1 bits on the stack. Then, when the SR contents are popped from the stack upon RETI, the **operating mode will be active mode (AM)**.

There are six operating modes that the software can configure:

- ❑ 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
 - ACLK remains active

- ❑ 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
 - ACLK remains active

- ❑ Low power mode 4 (LPM4); SCG1=X, SCG0=X, OscOff=1, CPUOff=1:
 - CPU is disabled
 - ACLK is disabled
 - MCLK is disabled
 - SMCLK is disabled
 - DCO oscillator is disabled
 - Crystal oscillator is stopped

Operating Modes for Low Power

More details

❑ **Low-Power Mode 0 and 1 (LPM0 and LPM1)**

- Low power mode 0 or 1 is selected if bit CPUOff in the status register is set. Immediately after the bit is set the CPU stops operation, and the normal operation of the system core stops.
- The operation of the CPU halts and all internal bus activities stop until an interrupt request or reset occurs.
- The system clock generator continues operation, and the clock signals MCLK, SMCLK, and ACLK stay active depending on the state of the other three status register bits, SCG0, SCG1, and OscOff.
- The 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.

❑ **Low-Power Modes 2 and 3 (LPM2 and LPM3)**

- Low-power mode 2 or 3 is selected if bits CPUOff and SCG1 in the status register are set.
- Immediately after the bits are set, CPU, MCLK, and SMCLK operations halt and all internal bus activities stop until an interrupt request or reset occurs.
- Peripherals that operate with the MCLK or SMCLK signal are inactive because the clock signals are inactive.
- Peripherals that operate with the ACLK signal are active or inactive according with the individual control registers and the module enable bits in the SFRs.
- All I/O port pins and the RAM/registers are unchanged.
- Wake up is possible by enabled interrupts coming from active peripherals or RST/NMI.

❑ Low-Power Mode 4 (LPM4)

- System Resets, Interrupts, and Operating Modes: in low power mode 4 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.

Principles of low power operation on the MSP

The User's Guide for each type of MSP430xx explains the principles needed to lower the power consumption of a design.

1. First, **minimize wasteful code execution**. This is the same idea as **improving speed performance** because every unnecessary instruction wastes a little bit of battery power. All of the techniques that **improve code efficiency will improve power efficiency**.
2. Increasing clock speed will not yield similar power savings because faster execution increases power consumption.
3. Similarly, unused **peripheral modules** on the processor should be **de-activated** to save power.
4. **Use interrupts** to handle events to allow the processor to stay in Low Power Mode 3 as much as possible.
5. By reducing the awake time of the processor, the average current consumption of the MSP430 can be reduced to levels approximately as low as LPM3 average, while maintaining the same functionality.