MSP430: Basic Clock System

- Low System Cost
- Low Power
- Variety of operating modes driven by application, software selectable
- Support for the Burst Mode – when activated system starts and reacts rapidly
- Stability over voltage and temperature
Basic Clock Systems-detail

- **Basic Clock Systems**
  - **Auxiliary Clock (ACLK)**
  - **Main System Clock (MCLK)**
  - **Sub-System Clock (SMCLK)**

### Digital Controlled Oscillator (DCO)
- **SCG0** - when set, turns off the DCO
- **SCG1** - when set, turns off the SMCLK
- **OSCOFF** - when set, turns off the LFXT1 crystal oscillator
- **CPUOFF** - when set, turns off the CPU

### Control Registers
- **DCOCTL**
- **BCSCTL1**
- **BCSCTL2**

### Basic operation
- After POC (Power Up Clear)
  - MCLK and SCLK are sourced by DCOCLK (approx. 800KHz)
  - ACLK is sourced by LFXT1 in LF mode
- Status register control bits SCG0, SCG1, OSCOFF, and CPUOFF configure the MSP430 operating modes and enable or disable portions of the basic clock module.
  - SCG1 - when set, turns off the SMCLK
  - SCG0 - when set, turns off the DCO generator
  - OSCOFF - when set, turns off the LFXT1 crystal oscillator
  - CPUOFF - when set, turns off the CPU
- **DCOCTL**, **BCSCTL1**, and **BCSCTL2** registers configure the basic clock module.
- The basic clock can be configured or reconfigured by software at any time during program execution.

### Low-power operation: An example
- ACLK can be configured to oscillate with a low-power 32,768-Hz watch crystal
- MCLK can be configured to operate from the on-chip DCO that can be only activated when requested by interrupt-driven events
- SMCLK can be configured to operate from either the watch crystal or the DCO, depending on peripheral requirements
LFXT1 Oscillator

- **LF mode: XTS = 0**
  - 32,768-Hz watch crystal in LF mode. A watch crystal connects to XIN and XOUT without any other external components.

- **HF mode: XTS = 1**
  - The high-speed crystal or resonator connects to XIN and XOUT and requires external capacitors on both terminals. These capacitors should be sized according to the crystal or resonator specifications.

- Software can disable LFXT1 by setting OSCOFF, if this signal does not source SMCLK or MCLK.

XT2 Oscillator

- Similar to LFXT1 in HF mode
- XT2OFF bit disables the XT2 oscillator if XT2CLK is not used for MCLK or SMCLK.

Digitally-Controlled Oscillator DCO

- Integrated ring oscillator with RC-type characteristics
  - Frequency varies with temperature, voltage, and from device to device.

- DCO frequency can be adjusted by software using the DCOx, MODx, and RSELx bits.
  - The digital control of the oscillator allows frequency stabilization despite its RC-type characteristics.

- After a PUC, an internal resistor is selected for the DCO generator.
  - RSELx=4, and DCOx=3 => start at a mid-range frequency

- MCLK and SMCLK are sourced from DCOCLK. Because the CPU executes code from MCLK, which is sourced from the fast-starting DCO, code execution begins from PUC in less than 6us.

Adjusting frequency

- **DCO frequency is determined by:**
  - The current injected into the DC generator by either the internal or external resistor defines the fundamental frequency.
  - DCO bit selects the internal or external resistor.
  - The three RSELx bits select one of eight nominal frequency ranges for the DCO.
  - The three DCOx bits divide the DCO range into 8 frequency steps, separated by approx. 10%.
  - The five MODx bits, switch between the frequency selected by the DCOx bits and the next higher frequency set by DCO=1.

Disabling DCO

- Software can disable DCO if not used for MCLK and SMCLK.
Oscillator and Clock Control Register

BCSCTL1 is affected by a valid PUC or POR condition.

<table>
<thead>
<tr>
<th>Bit</th>
<th>XT2Off</th>
<th>XT5V</th>
<th>DIVA.1</th>
<th>DIVA.0</th>
<th>XTS</th>
<th>DIVA.1</th>
<th>Rsel.5</th>
<th>Rsel.4</th>
</tr>
</thead>
<tbody>
<tr>
<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>
</tbody>
</table>

Bit0 to Bit2: The internal resistor is selected in eight different steps.

Rsel.0 to Rsel.2: The value of the resistor defines the nominal frequency. The lowest nominal frequency is selected by setting Rsel=0.

Bit3, XT5V: XT5V should always be reset.

Bit4 to Bit5: The selected source for ACLK is divided by:

- DIVA = 0: 1
- DIVA = 1: 2
- DIVA = 2: 4
- DIVA = 3: 8

Bit6, XTS: The LFXT1 oscillator operates with a low-frequency or with a high-frequency crystal.

- XT5V = 0: The low-frequency oscillator is selected.
- XT5V = 1: The high-frequency oscillator is selected.

The oscillator selection must meet the external crystal’s operating condition.

Bit7, XT2Off: The XT2 oscillator is switched on or off:

- XT2Off = 0: the oscillator is on
- XT2Off = 1: the oscillator is off if it is not used for MCLK or SMCLK.

BCSCTL2 is affected by a valid PUC or POR condition.

<table>
<thead>
<tr>
<th>Bit</th>
<th>SELM.1</th>
<th>SELM.0</th>
<th>DIVM.1</th>
<th>DIVM.0</th>
<th>SELS</th>
<th>DIVS.1</th>
<th>DIVS.0</th>
<th>DCOR</th>
</tr>
</thead>
<tbody>
<tr>
<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>
</tbody>
</table>

Bit0, DCOR: The DCOR bit selects the resistor for injecting current into the dc generator. Based on this current, the oscillator operates if activated.

- DCOR = 0: Internal resistor on, the oscillator can operate. The fail-safe mode is on.
- DCOR = 1: Internal resistor off, the current must be injected externally if the DCO output drives any clock using the DDCCLK.

Bit1, Bit2: The selected source for SMCLK is divided by:

- DIVS.1 .. DIVS.0: DIVS = 0: 1
- DIVS = 1: 2
- DIVS = 2: 4
- DIVS = 3: 8

Bit3, SELS: Selects the source for generating SMCLK:

- SELS = 0: Use the DCCCLK
- SELS = 1: Use the XT2CLK signal (in three-oscillator systems) or LFXT1CLK signal (in two-oscillator systems)

Bit4, Bit5: The selected source for MCLK is divided by DIVM.0 .. DIVM.1

- DIVM = 0: 1
- DIVM = 1: 2
- DIVM = 2: 4
- DIVM = 3: 8

Bit6, Bit7: Selects the source for generating MCLK:

- SELM.0 .. SELM.1: SELM = 0: Use the DCCCLK
- SELM = 1: Use the XT2CLK (x13x and x14x devices) or Use the LFXT1CLK (x11x devices)
- SELM = 2: Use the LFXT1CLK (x11x devices)
- SELM = 3: Use the LFXT1CLK

Range (RSELx) and Steps (DCOx)
External Resistor

- The DCO temperature coefficient can be reduced by using an external resistor ROSC to source the current for the DC generator.
- ROSC also allows the DCO to operate at higher frequencies.
- Internal resistor nominal value is approximately 200 kOhm, hence the DCO can operate up to approximately 10 MHz.
- External ROSC of approximately 100 kOhm allows the DCO to operate up to approximately 10 MHz.

Basic Clock Systems-software FLL idea

- Basic Clock DCO is an open loop - close with SW+HW
- A reference frequency e.g. ACLK or 50/60Hz can be used to measure DCOCLK's deviation.
- Initialization or Periodic software set and stabilizes DCOCLK over reference clock
- DCOCLK is programmable 100kHz - 5MHz, stable over voltage and temperature

Basic Clock Systems-software FLL implementation

- Example: Set DCOCLK= 1228800, ACLK= 32768
- ACLK/4 captured on CCI2B, DCOCLK is clock source for Timer_A
- Comparator2 HW captures SMCLK (1228800Hz) in one ACLK/4 (8192Hz) period
- Target Delta = 1228800/8192 = 150

```assembly
CCI2BInt … ; Compute Delta
cmp #150, Delta ; Delta = 1228800/8192
jlo IncDCO ; JMP to IncDCO
DecDCO dec &DCOCTL ; Decrease DCOCLK
reti
IncDCO inc &DCOCTL ; Increase DCOCLK
reti
```

Basic Clock Systems-DCO TAPS

- DCOCLK frequency control
  - nominal - injected current into DC generator
  - 1) internal resistors Rsel2, Rsel1 and Rsel0
  - 2) an external resistor at Rosc (P2.5/11x)
- Control bits DCO0 to DCO2 set fDCO tap
- Modulation bits MOD0 to MOD4 allow mixing of fDCO and fDCO+1 for precise frequency generation

<table>
<thead>
<tr>
<th>Example</th>
<th>Selected:</th>
</tr>
</thead>
<tbody>
<tr>
<td>f3:</td>
<td>1000kHz</td>
</tr>
<tr>
<td>f4:</td>
<td>943kHz</td>
</tr>
<tr>
<td></td>
<td>1042kHz</td>
</tr>
<tr>
<td>Frequency Cycle time</td>
<td>1000 nsec</td>
</tr>
<tr>
<td>MOD=19</td>
<td></td>
</tr>
</tbody>
</table>

Synchronization of clock signals

- When switching MCLK and SMCLK from one clock source to another
  - avoid race conditions
  - The current clock cycle continues until the next rising edge
  - The clock remains high until the next rising edge of the new clock
  - The new clock source is selected and continues with a full period
Basic Clock Systems-Examples

- How to select the Crystal Clock

```c
void selectclock(void) {
    IFG2=0; /* reset interrupt flag register 2 */
    IFG1=0; /* reset interrupt flag register 1 */
    BCSCTL1|=XTS; /*attach HF crystal (4MHz) to XIN/XOUT */
    do {
        /*wait in loop until crystal is stable*/
        IFG1&=~OFIFG;
    }while(OFIFG&IFG1);
    Delay();
    IFG1&=~OFIFG; /*Reset osc. fault flag again*/
}
```

- How to select a clock for MCLK

```c
BCSCTL2=SELM0+SELM1; /*Then set MCLK same as LFXT1CLK*/
TACTL=TASSEL0+TACLR+ID1; /*USE ACLK/4 AS TIMER_A INPUT CLOCK (1MHz)*/
```

Adjusting the Basic Clock

The control registers of the Basic Clock are under full software control. If clock requirements other than those of the default from PUC are necessary, the Basic Clock can be configured or reconfigured by software at any time during program execution.

- ACUI2GEN from LFXT1 crystal, resonator, or external-clock source and divided by 1, 2, 4, or 8. If no LFXTCLK clock signal is needed in the application, the OscOff bit should be set in the status register.

- SCLKGEN from LFXTCLK, DCOCLK, or XT2CLK (x13x and x14x only) and divided by 1, 2, 4, or 8. The SCG1 bit in the status register enables or disables SCLK.

- MCLKGEN from LFXTCLK, DCOCLK, or XT2CLK (x13x and x14x only) and divided by 1, 2, 4, or 8. When set, the CPUOff bit in the status register enables or disables MCLK.

- DCOCLK frequency is adjusted using the RSEL, DCO, and MOD bits. The DCOCLK clock source is stopped when not used, and the dc generator can be disabled by the SCG0 bit in the status register.

- The XT2 oscillator sources XT2CLK (x13x and x14x only) by clearing the XT2Off bit.

Interrupt Service Routines

- Interrupt Routine declaration

```c
// Func. declaration
Interrupt[int_vector] void myISR (Void);

Interrupt[int_vector] void myISR (Void) {
    // ISR code
}
```

EXAMPLE

```c
Interrupt[TIMERA0_VECTOR] void myISR (Void);
Interrupt[TIMERA0_VECTOR] void myISR (Void) {
    // ISR code
}
```

MSP430: Watchdog Timer

General

The primary function of the watchdog-timer module (WDT) is to perform a controlled-system restart after a software problem occurs. If the selected time interval expires, a system reset is generated. If the watchdog function is not needed in an application, the module can work as an interval timer, to generate an interrupt after the selected time interval.

Features of the Watchdog Timer include:

- Eight software-selectable time intervals
- Two operating modes: as watchdog or interval timer
-Expiration of the time interval in watchdog mode, which generates a system reset; or in timer mode, which generates an interrupt request
-Safeguards which ensure that writing to the WDT control register is only possible using a password
-Support of ultralow-power using the hold mode

Watchdog/Timer two functions:

- SW Watchdog Mode
- Interval Timer Mode
The WDTIE bit is used to enable or disable the interrupt from the Watchdog Timer when in interval-timer mode. Also, the GIE bit enables or disables the interrupt from the Watchdog Timer when it is being used in interval-timer mode.

Table: WDTCNT Taps

<table>
<thead>
<tr>
<th>SSEL</th>
<th>IS2</th>
<th>IS0</th>
<th>Interval [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.064 &amp; SMCLK x 2^1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.5 &amp; SMCLK x 2^0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1.0 &amp; ACLK x 2^1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8 &amp; SMCLK x 2^3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>16.0 &amp; ACLK x 2^4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32 &amp; SMCLK x 2^5</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>250 &amp; ACLK x 2^8</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1000 &amp; ACLK x 2^10</td>
</tr>
</tbody>
</table>

Bit 2: The SSEL bit selects the clock source for WDTCNT. SSEL = 0: WDTCNT is clocked by SMCLK. SSEL = 1: WDTCNT is clocked by ACLK.

Bit 3: Counter clear bit. In both operating modes, writing a 1 to this bit restarts the WDTCNT at 00000h. The value read is not defined.

Bit 4: The TMSEL bit selects the operating mode: watchdog or timer. TMSEL = 0: Watchdog mode TMSEL = 1: Interval-timer mode

Bit 5: The NMI bit selects the function of the RST/NMI input pin. It is sensitive.

The Watchdog Timer Interrupt Function

The Watchdog Timer (WDT) uses two bits in the SFRs for interrupt control.

The WDT interrupt flag (WDTIFG) (located in SFR1.0, initial state is reset)

The WDT interrupt enable (WDTIE) (located in IE1.0, initial state is reset)

When using the watchdog mode, the WDTIFG flag is used by the reset interrupt service routine to determine if the watchdog caused the device to reset. If the flag is set, then the Watchdog Timer initiated the reset condition (either by timing out or by a security key violation). If the flag is cleared, then the PUC was caused by a different source. See chapter 3 for more details on the PUC and POR signals.

When using the Watchdog Timer in interval-timer mode, the WDTIFG flag is set after the selected time interval and a watchdog interval-timer interrupt is requested. The interrupt vector address in interval-timer mode is different from that in watchdog mode. Also, the GIE bit enables or disables the interrupt from the Watchdog Timer when it is being used in interval-timer mode.

Watchdog Timer-Registers

- **WDTCTL**
  - **bits**: 0124h
  - **function**: 16-bit control register with write protection

- **WDTCNT**
  - **function**: 16-bit up-counter that is not directly accessible by software.

- **WDT**
  - **function**: Holds the last value read from a specific address until the hold bit is reset and the operation continues. It is cleared by the PUC signal.

- **NMI**
  - **function**: A falling edge triggers an NMI interrupt.

- **CAUTION**: Changing the NMIES bit with software can generate an NMI interrupt.

Watchdog Timer-Mode

- **Setting WDTCTL register bit TMSEL to 1 selects the timer mode. This mode provides periodic interrupts at the selected time interval. A time interval can also be initiated by writing a 1 to bit CNTCL in the WDTCTL register.**

- **When the WDT is configured to operate in timer mode, the WDTIFG flag is set after the selected time interval, and it requests a standard interrupt service. The WDT interrupt flag is a single-source interrupt flag and is automatically reset when it is serviced. The enable bit remains unchanged. In interval-timer mode, the WDT interrupt-enable bit and the GIE bit must be set to allow the WDT to request an interrupt. The interrupt vector address in timer mode is different from that in watchdog mode.**
Watchdog Timer-Examples

- How to select timer mode
  ```c
  /* WDT is clocked by fACLK (assumed 32Khz) */
  WDTCL=WDT_ADLY_250; // WDT 250MS/4 INTERVAL TIMER
  IE1 |=WDTIE;        // ENABLE WDT INTERRUPT
  ```

- How to stop watchdog timer
  ```c
  WDTCTL=WDTPW + WDTHOLD ;  // stop watchdog timer
  ```

Assembly programming

```c
WDT_key .equ    05A00h  ; Key to access WDT
WDTStop mov #(WDT_Key+80h),&WDTCTL; Hold Watchdog
WDT250 mov #(WDT_Key+1Dh),&WDTCTL; WDT, 250ms Interval
```

Power as a Design Constraint

Power becomes a first class 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

Dynamic Power Consumption

- \( C \) - Total capacitance seen by the gate’s outputs
- \( V \) - Supply voltage
- \( f \) - clock frequency
- \( A \) - Activity of gates
- \( \tau \) - Transition time

\[
\text{Power due to short-circuit current during transition} = AV_{\text{short}} \tau
\]

Reducing Dynamic Power
1) Reduce \( V \) 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

Short-circuit Power Consumption

- \( I_{\text{short}} \)
- \( \tau \)
- Finite slope of the input signal causes a direct current path between \( V_C \) 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 risefall time of the input and output signals
Leakage Power

\[ V_{\text{leak}} \]

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

CMOS Power Equations

\[ P = ACV^2f + \tau V_{\text{short}}f + V_{\text{leak}} \]

- Reduce the supply voltage, \( V \)
- \( f_{\text{max}} \times \frac{(V - V_t)^2}{V} \)
- \( I_{\text{leak}} \propto \exp(-qV_t / kT) \)

How can we reduce power consumption?

- Dynamic power consumption
  - 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?

Average power consumption

- Dynamic power supply current
  - Set of modules that are periodically active
  - Typical situation – real time cycle \( T \)
- \( I_{\text{ave}} = \int I_{\text{cc}}(t)dt / T \)
- In most cases \( I_{\text{ave}} = \sum I_i \times t_i / T \)

Low-Power Concept: Basic Conditions for Burst Mode

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

<table>
<thead>
<tr>
<th>Measure</th>
<th>Process step</th>
<th>Real-Time Clock</th>
<th>LCD/Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measure</td>
<td>Calculate</td>
<td>RTC</td>
<td>Display</td>
</tr>
<tr>
<td>ADC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3mA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5mA</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The sleep current dominates the current consumption!

Battery Life

- Battery Capacity BC – [mAh]
- Battery Life
  - \( BL = BC / I_{\text{ave}} \)
- In the previous example, standard 800 mAh batteries will allow battery life of:
  - \( BL = 750 \, \text{mAh} / 2.1 \, \mu\text{A} = 44 \, \text{years} !!! \)
- Conclusion:
  - Power efficient modes
  - Interrupt driven system with processor in idle mode
Low power - features

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

Reducing power consumption

- Logic
  - Clock tree (up to 30% of power)
  - Clock gating (turn off branches that are not used)
  - Half frequency clock (both edges)
  - Half swing clock (half of Vcc)
  - Asynchronous logic
    - completion signals
    - testing
- Architecture
  - Parallelism (increased area and wiring)
  - Speculation (branch prediction)
  - Memory systems
    - Memory access (dynamic)
    - Leakage
    - Memory banks (turn off unused)
- Buses
  - 32-64 address/data, (15-20% of power)
  - Gray Code, Code compression

Reducing power consumption #2

- Operating System
  - Finish computation "when necessary"
  - Scale the voltage
    - Application driven
    - Automatic
  - System Architecture
    - Power efficient and specialized processing cores
    - A "convergent" architecture
    - Trade-off
      - CPUOff, OscOff, SCG0, and SCG1.
  - Other issues
    - Leakage current – Thermal runaway
    - Voltage clustering (low Vthreshold for high speed paths)

Operating Modes-General

The MSP430 family was developed for ultra-low-power applications and uses different levels of operating modes. The MSP430 operating modes, give advanced support to various requirements for ultra-low power and ultra-low 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 instruction returns operation to the mode that was selected before the interrupt event.

The ultra-low power system design which uses complementary metal-oxide semiconductor (CMOS) technology, takes into account three different needs:

1. The desire for speed and data throughput despite conflicting needs for ultra-low power
2. Minimization of individual current consumption
3. Limitation of the activity state to the minimum required by the use of low power modes

Low power mode control

There are four bits that control the CPU and the main parts of the operation of the system clock generator:

- CPUOff
- OscOff
- SCG0, and
- SCG1.

These four bits support discontinuous active mode (AM) requests, to limit the time period of the full operating mode, and are located in the status register. The major advantage of including the operating mode bits in the status register is the 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.

Operating Modes-General

Another program flow may be selected by manipulating the data stored on the stack or the stack pointer. 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:

- 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.
- Return from interrupt
  - Two different modes are available to return from the interrupt service routine and continue the flow of operation:
    - Return with low-power mode bits set. When returning from the interrupt, the program counter points to the next instruction. The instruction pointed to is not executed, since the restored low power mode stops CPU activity.
    - 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 or SCG0 or SCG1 in the status register. To use this mode, the interrupt service routine must reset the OscOff, CPUOff, SCG0, and SCG1 bits in the stack. Then, when the SR contents are popped from the stack upon RETI, the operating mode will be active mode (AM).
Operating Modes - Software configurable

There are six operating modes that the software can configure:

- **Active mode:**
  - SCG1=0, SCG0=0, OSCOFF=0, CPUOFF=0:
    - CPU is disabled
    - MCLK is disabled
    - SMCLK and ACLK remain active
    - DCO is active
    - The SR is pushed onto the stack

- **Low power mode 0 (LPM0):**
  - SCG1=0, SCG0=0, OSCOFF=0, CPUOFF=1:
    - CPU is disabled
    - MCLK is disabled
    - SMCLK and ACLK remain active
    - DCO is active

- **Low power mode 1 (LPM1):**
  - SCG1=0, SCG0=0, OSCOFF=0, CPUOFF=1:
    - CPU is disabled
    - MCLK is disabled
    - SMCLK is disabled
    - DCO oscillators are disabled
    - DCO oscillator automatically disabled because it is not needed for MCLK or SMCLK

- **Low power mode 2 (LPM2):**
  - SCG1=1, SCG0=0, OSCOFF=0, CPUOFF=1:
    - CPU is disabled
    - MCLK is disabled
    - SMCLK and ACLK remain active
    - DCO oscillator is disabled

- **Low power mode 3 (LPM3):**
  - SCG1=1, SCG0=0, OSCOFF=1, CPUOFF=1:
    - CPU is disabled
    - MCLK is disabled
    - SMCLK is disabled
    - DCO oscillators are disabled

- **Low power mode 4 (LPM4):**
  - SCG1=1, SCG0=1, OSCOFF=0, CPUOFF=1:
    - CPU is disabled
    - MCLK is disabled
    - SMCLK is disabled
    - DCO oscillator is disabled

- **Exit from Low Power Mode:**
  - **LPM0 exit:**
    - the SR is popped from the stack
  - **LPM1 exit:**
    - the SR is popped from the stack
    - the SR is pushed onto the stack
  - **LPM2 exit:**
    - the SR is popped from the stack
    - the SR is pushed onto the stack
  - **LPM3 exit:**
    - the SR is popped from the stack
    - the SR is pushed onto the stack
  - **LPM4 exit:**
    - the SR is popped from the stack
    - the SR is pushed onto the stack

- **Clock Signals:**
  - MCLK, SMCLK, ACLK

- **Low-Power Modes 2 and 3 (LPM2 and LPM3):**
  - The peripherals are enabled or disabled with their individual control registers in the SFRs.
  - The peripherals that are active in LPM2 and LPM3 are:
    - I/O ports
    - RAM/registers
  - Wake up is possible through enabled interrupts.

- **Interrupt Service Routine:**
  - The program continues here if the CPUOff bit is set during the interrupt service routine.

- **Low-Power Mode 4 (LPM4):**
  - SCG1=1, SCG0=1, OSCOFF=1, CPUOFF=1:
    - CPU is disabled
    - MCLK is disabled
    - SMCLK is disabled
    - DCO oscillators are disabled
    - DCO’s dc-generator is disabled
    - Crystal oscillator is stopped

- **Operating Modes - Low Power Mode in details:**
  - **Low-Power Mode 0:**
    - SCG1=0, SCG0=0, OscOff=0, CPUOff=0:
      - CPU is disabled
      - MCLK is disabled
      - SMCLK and ACLK remain active
      - DCO is active
  - **Low-Power Mode 1:**
    - SCG1=0, SCG0=1, OscOff=0, CPUOff=1:
      - CPU is disabled
      - MCLK is disabled
      - SMCLK and ACLK remain active
      - DCO is disabled
  - **Low-Power Mode 2:**
    - SCG1=1, SCG0=0, OscOff=0, CPUOff=1:
      - CPU is disabled
      - MCLK is disabled
      - SMCLK and ACLK remain active
      - DCO is disabled
  - **Low-Power Mode 3:**
    - SCG1=1, SCG0=1, OscOff=1, CPUOff=1:
      - CPU is disabled
      - MCLK is disabled
      - SMCLK is disabled
      - DCO is disabled

- **Operating Modes - Software configurable:**
  - **SCG1:**
    - 0 = SMCLK, ACLK, MCLK, DCO are enabled
    - 1 = SMCLK, ACLK, MCLK, DCO are disabled
  - **SCG0:**
    - 0 = SMCLK, ACLK, MCLK, DCO are enabled
    - 1 = SMCLK, ACLK, MCLK, DCO are disabled
  - **OSCOFF:**
    - 0 = SMCLK, ACLK, MCLK, DCO are enabled
    - 1 = SMCLK, ACLK, MCLK, DCO are disabled
  - **CPUOFF:**
    - 0 = SMCLK, ACLK, MCLK, DCO are enabled
    - 1 = SMCLK, ACLK, MCLK, DCO are disabled

- **Operating Modes - Low Power Mode in details:**
  - **Low-Power Mode 4:**
    - SCG1=1, SCG0=1, OscOff=1, CPUOff=1:
      - CPU is disabled
      - MCLK is disabled
      - SMCLK is disabled
      - DCO oscillators are disabled

- **Operating Modes - Low Power Mode in details:**
  - **CPE/EE 421/521 Microcomputers 65**

- **Operating Modes - Low Power Mode in details:**
  - **CPE/EE 421/521 Microcomputers 66**

- **Operating Modes - Examples:**
  - **CPE/EE 421/521 Microcomputers 65**

- **Operating Modes - C Examples:**
  - **CPE/EE 421/521 Microcomputers 66**