Goals:

- Learn how to operate in LPM (Low Power Mode).
- Learn how to implement Port Interrupts in your designs.
- Create an 8-bit BCD (Binary Coded Decimal) Counter. The counter should have three functions implemented with three buttons: increment, decrement and reset to zero. Current value of counter should be outputted to an SSD (Seven Segment Display); 8 LEDs may be used instead.
- Counter should count from 0 – 99.
- Design must be LPM and Interrupt Driven.

Bonus:

Write your Binary to Decimal encoding algorithm/subroutine in 10 lines of code or less. +10
Write your own BCD to SSD subroutine to output the current value to the SSD (do not use an external BCD decoder chip). +20

Pre Lab Questions:

- Of the 5 LPM modes, which Low Power Mode did you choose to use? Did it matter; why or why not?
- Which port did you use for your inputs; outputs?
- Did you use an infinite loop; why or why not?
Today’s engineering designs are moving towards energy efficient designs. This means we must learn to develop systems that are low power and interrupt driven. Read below to find out more.

**Low Power Mode**

**LPM** allows us to operate at a much lower than our already minuscule current consumption rate. See the graph below taken from Section 2.3 – Operating Modes from the MSP430x2xx Family User's Guide (Rev. I):

![Graph showing current consumption at different operating modes](image)

In **LPM**, various crystals, clocks and the CPU are disabled. You must take care that the peripheral you are relying on is not dependent on any of these timing sources.
Below is a table summarizing the different Low Power Modes:

<table>
<thead>
<tr>
<th>MODE</th>
<th>MACROS</th>
<th>CONTROL BITS</th>
<th>CPU &amp; CLOCK STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Function</td>
<td>SCG1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td></td>
<td>--</td>
<td>0</td>
</tr>
<tr>
<td>LPM0</td>
<td>LPM0</td>
<td>LPM0_bits</td>
<td>0</td>
</tr>
<tr>
<td>LPM1</td>
<td>LPM1</td>
<td>LPM1_bits</td>
<td>0</td>
</tr>
<tr>
<td>LPM2</td>
<td>LPM2</td>
<td>LPM2_bits</td>
<td>1</td>
</tr>
<tr>
<td>LMP3</td>
<td>LPM3</td>
<td>LPM3_bits</td>
<td>1</td>
</tr>
<tr>
<td>LPM4</td>
<td>LPM4</td>
<td>LPM4_bits</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1 - Low Power Mode Details

Note: Obviously, we cannot use a timer interrupt to wake from a LPM in which we have a clock disabled as that will make the associated timer interrupt inactive.

Port Interrupts

Just like with the previous Port Registers we learned about, Port Interrupt Registers are 8-bit and work the same way as far as bits corresponding to certain pins. Below is a description of each:

- **PxIE** - Interrupt Enable; 0 – Disable; 1 – Enable
- **PxIES** - Interrupt Edge Select; 0 – Rising Edge; 1 – Falling Edge
- **PxIFG** - Interrupt Flag; 0 – Cleared; 1 – Set

Ports 1 and 2 are never disabled when entering a LPM. If you want to save even more power, you must set all output pins to inputs and ground them with the internal resistor (Pulldown Configuration). This means that we will (almost) always be able to use a Port Interrupt to wake from LPM.

For more information on Low Power Mode and Port Interrupts, see the Interrupts Guide.
8-bit BCD (Binary Coded Decimal) Counter

In this lab, we will be designing an 8-bit BCD (Binary Coded Decimal) Counter. Simply put, every nibble in binary represents a power of tens place in decimal. See the table below for a better understanding:

<table>
<thead>
<tr>
<th>Decimal Number</th>
<th>MSN (Tens)</th>
<th>LSN (Ones)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>0000</td>
<td>0001</td>
</tr>
<tr>
<td>2</td>
<td>0000</td>
<td>0010</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>98</td>
<td>1001</td>
<td>1000</td>
</tr>
<tr>
<td>99</td>
<td>1001</td>
<td>1001</td>
</tr>
</tbody>
</table>

For example, if we wanted to encode the decimal number 47, we would use 01000111.

Notes:

- It is possible to enter LPM during an ISR (Interrupt Service Routine). However, you must remember reset the GIE bit in the SR. The easiest way to do this is to re-enable general interrupts right before entering LPM. See the Interrupts Guide for more details.
- You may enter a different LPM or return to Active Mode after an Interrupt.
- Ports do not rely on any timing source.
- LPM allows us to neglect using an infinite while loop in our main program.
- Other than code to setup the necessary registers and enter LPM, there should be no other code in the main program.
- If your micro is currently in LPM, you will not be able to see where in code you are until an interrupt is triggered.
Because we are using a **LPM** and **Interrupt Driven** design, we will not be needing an infinite while loop. Normally we would use an empty infinite while loop as such:

```c
while(1){};
```

Instead, we will setup our **Port Registers (I/O and Interrupt)** and then immediately enter **LPM**. We will wake the micro with a **Port Interrupt**, manipulate the **BCD Counter** value and re-enter **LPM**. See the flowchart below for help:

Remember to use a **Global Variable** to hold the binary/decimal counter value.
Lab Notebook Questions:

• Explain why it is preferable to use interrupts instead of polling.
• Is polling completely avoidable?
• Are infinite loops in the main program completely avoidable?
• What happens when you go into a LPM; what happens when you return?
**Low Power Mode and Port Interrupts**

**Frequently Asked Questions**

*Do all pins go to the same ISR?*

Yes. Both Ports 1 and 2 have their respective ISRs and each of their pins source (trigger) these ISRs.

*So then how can I know what pin sourced the ISR?*

If you have multiple possible sources for the ISR, you must read the PxIFG register to see what bit is set; the set bit will tell you which pin sourced the ISR.

*How do I connect my switch!?*

Checkout the diagram below if you are having trouble connecting your switch: