Sleep

My name is Alan Hawse and this is PSoC 101. The last two projects in this series of lessons discuss low power support. I am also going to show you the pre-populated schematic templates that I told you about way back in lesson one. These templates are schematic files that already include a lot of MCU-like functionality and they save you the task of finding and adding and configuring things like UARTs, the ADC, timers, PWMs, and so on. The templates are all customized for the specific part number your project is targeting so they all build without error straight out of the box.

In this lesson I am going to show you the power consumption savings you can make by switching the part into sleep mode. In this mode the CPU is turned off but all the other hardware still runs. Many of the lessons we did in this series would run perfectly well if you entered sleep mode rather than executing an empty infinite loop.

You will need a digital multimeter to test the power savings in this lesson. The project will run for 10s in active mode then sleep for 10s until a Timer wakes us up again. Start by creating a new project for your target device. Remember to pick the right family. Then choose the pre-populated schematics, instead of the empty one you have been using so far.

You'll immediately notice that the schematic contains communication components that you can quickly edit to set the right parameters. We are going to use the UART but there is no need to change anything at all in its customizer dialog. There are also extra tabs in the schematic file for digital, ADC, CapSense, and so on. On the far right is a tab called My Design which is just a handy place for you to start your custom design work, but you can create as many new tabs as you like.

In this project we only need the Communication and Digital tab so go ahead and disable all the others. You can turn them back on if you ever want to extend this design with new functionality. In the digital tab you will recognize the PWMs and Timers. You will be using the Timer. If you want a little bit of space you can delete the comments. If you like you can delete the PWMs. Then delete the switch and the wires to the Timer because we are controlling the component from firmware. Slow the clock down to 1kHz so the math to create a ten second timer is really easy. In the Timer, make it one shot with a time of 10000 (10s), set the interrupt on terminal count, and clear the inputs that used to be wired to the switch pin. You now have a timer that will run for 10 seconds and then fire an interrupt.

In the DWR file, choose appropriate pins for the UART on your kit then generate the application to build the API functions.

You have done most of this before so just start your UART and register a Timer ISR using StartEX. Above main() write the ISR itself. All it needs to do is clear the interrupt. The timer interrupt will simply wake the part from sleep mode and execute code from the point where it was put to sleep. In your main loop use the CyDelay function to wait long enough to take readings for active mode current consumption and then print a helpful message to the UART, start the one-shot Timer, and go to sleep with the CySysPmSleep function. The device will enter sleep mode and stay there for 10 seconds, plenty long enough to measure the current. When it wakes up just print another useful message.

All of the Pioneer kits allow you to remove a jumper and attach a multimeter to the exposed pins. Check the psoc101 web page for the specific instructions for your kit. When you run the project you should see that turning off the CPU reduces the power consumption of the system by about 50%.

For your project I'd like you to try configuring an input pin on the SW2 button and use that to generate an interrupt that will cause an exit from sleep mode. You have configured the pin before – remember to set the drive mode and enable interrupts on a falling edge – and map it to the SW2 switch in the DWR file.

As always you are welcome to email me at alan_hawse@cypress.com.