Software Output Pins

My name is Alan Hawse and this is PSoC 101. Let's get working on the kit. The first thing we need to do is create a new project in PSoC Creator. In the dialog I can give my project a name and choose the target device. Because I am using the 042 kit I need to select the PSoC 4100 / PSoC 4200 design and then choose a template for the project. In this drop-down I can either choose an empty schematic or one that is pre-populated with all the usual MCU functions, like UARTs and an ADC. Because we're learning from scratch we will use the empty template for now. After clicking OK you will be magically transported to the blank schematic page. If you see blocks with titles like UART and I2C then you forgot to choose the empty template, go back and try again.

If you are using PSoC Creator 3.3 the new project dialog is a little different.

This project is going to be an led blinky, where we flash LEDs from firmware- the hello world of embedded programming. On the right-hand side of the screen is our Component Catalog with more than 100 preverified special functions. In the search box at the top enter the string p-i-n and you will be presented with the matching components. We need an output pin and you simply drag and drop it into your design. To set up the pin you just double-click to open its customizer dialog. Each component has a customizer that allows you to configure its functionality. In the dialog I will give the pin a name I can remember – Pin underscore green – and I'll turn off the HW terminal because we're going to drive this pin from firmware. I'll show you how to control pins from the PSoC hardware in a future lesson. You can see the complete index of lessons on www.cypress.com/psoc101.

We are going to drive an LED from this pin and, because the LEDs on the Pioneer kits are active-low, meaning they sink current to light the LED. I'll set the initial state to high so that the LED is off. Press OK to commit the changes and close the dialog. The LED on these kits are 3-color RGBs and so I'll make 2 copies of this component and rename them to blue and red respectively.

Now, I need to choose the physical address of the pins. That happens in the resources file double click to open it. Here you see the pins that we added to the schematic. I can drag each entry onto the picture or choose the pin from a drop-down menu.

My hardware is now configured. I can build this design and it will generate helpful C language functions to control the pins. These functions are placed in the Generated_Source folder. You can see them for each of the pins I placed.

In the main.c file I can add a little code for the application. All the set up is handled by the tool so I can go straight to using the generated functions to toggle the red pin. There is a write function and a read function. For this project if I read the pin state and write its inverse I will toggle the output. There is a handy delay-loop function called CyDelay which allows me to do nothing for a number of milliseconds. I’ll delay for 500 milliseconds, or half a second.

I need to build again and then program my board. I have already plugged the Pioneer kit into my USB port and so I’ll just press the program button. The tool knows I made edits in main.c and so it automatically saves the file and re-builds the firmware before it programs the board. Now you can see the LED blinking on and off once per second.

See if you can reproduce this project for yourself. Then try to modify it a little. You can change the color of the LED you blink and write a little C code so that you vary the duty cycle, making the LED stay on for longer than it is off.

As always you are welcome to email me at alan_hawse@cypress.com with your comments, suggestions, criticisms and questions.