Interrupts

My name is Alan Hawse and this is PSoC 101. In this lesson I am going to show you how to generate and handle an interrupt from the PSoC hardware. In this project you will attach an input pin to an interrupt, then use an Interrupt Service Routine to toggle the state of an LED. As before, make a copy of the input pin project first. I am going to make the input pin generate an interrupt. In the customizer navigate to the Input tab. Then select the interrupt to be on a falling edge. That corresponds to a press of the button because the pioneer kit switch is active low. You'll see an irq terminal appear on the bottom of the pin. Now, search for an interrupt component and drag that into the schematic. Wire it up to the terminal and give it a sensible name.

That is all we need to do in the design. The rest of the project is C code and to make that easier I'll build the project to have PSoC creator generate the APIs. The best way to become familiar with component APIs is to look in their datasheet. Every component has one and it can be accessed from many places in the PSoC Creator GUI. I usually just right-click on the component but you can also do it with the button in the customizer or from the Component Catalog. There is a section in the datasheet called the Application Programming Interface. For the ISR component you will see there is a Start and a StartEx function to install the handler and enable the interrupt. You can use either of these. The Start function requires you to add code in the generated source, which can be tricky if you are not used to it. I recommend that you use the StartEx function as it allows you to provide the address of your own handler.

First, I'll write the interrupt handler. I'll put it above main so I do not need to set up a forward declaration. We provide a compiler-agnostic macro to help you set up portable interrupt service routines. It is called CY_ISR and I invoke it with the name of my handler routine. The rest of the code is just the code for the ISR. I want to toggle the red LED on every button press so I'll write the code to read, invert and write back the LED pin's value. I also need to clear the interrupt or the ISR will keep on firing forever. The interrupt is generated from the pin so I'll open the pin datasheet and look at its API for a suitable function. It is called ClearInterrupt so I will call that in my ISR. Notice how the tool predicts your typing and helps you to get the correct API call.

After creating the handler I will need to install it by calling the StartEx function with the name of the function I just created.

You'll have noticed in the template main.c file that there is a call to CyGlobalIntEnable – this is a macro to turn on interrupts and so you need to make sure that line is not commented out to make your project work.

When I program the kit I can toggle the pin by pressing the switch. Sometimes I find that it ignores my presses though. This is because the switch bounces against its contact and generates multiple interrupts. We can fix that by adding a debouncer component to the circuit but we do not need to do that to get the hang of interrupts. When you make this project try controlling two LEDs from a single ISR – as in the previous lesson, turn one on and the other off.

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