Sidebar: Creating a Custom I²C Device in 15 Minutes

An embedded design tool that by default satisfies two of the four rules for successful I²C design in Part 1 in this series (Use a standardized register-based protocol and Follow an established set of design practices) PSoC Express allows the creation of a custom I²C slave in a matter of minutes.

After a high-level description of the device is entered and the PSoC for the end result is chosen, the tool generates the necessary firmware and provides a description of it in a custom-generated datasheet, complete with I²C register map and protocol description.

For example to create a 4-voltage monitor that indicates on an LED when all 4 voltages are within a specified limit, this tool uses a 4-step design process. First, select the devices wanted as inputs, outputs and communications interfaces. Then define for each output what its behavior is (called the transfer function for that device). Third, using the built-in behavioral simulator, verify the transfer functions are correct. Lastly, choose build to select the PSoC device (what family and what footprint) and define specific pin assignments if desired.

This can be done by dragging the “Input” icon onto the screen and selecting a voltage, 0-2600mV from the catalog that appears. Repeat this four times to add four voltage elements. Let’s call the voltages Volt1, Volt2, Volt3, and Volt4.

Now add the LED for power-good indication, by dragging the “Output” icon onto the screen and selecting an On/Off LED from the catalog, naming it PowerGood.
The next step is to define the behavior. To do this turn **PowerGood** LED on when all four voltages are within a specified limit. To do the limit checking on the voltages, add a “**Valuator**” icon to the screen and select “**SetPointRegion**” as the transfer function type.

Then select the voltage to check and specify the upper and lower limits. Create one **SetPoint Region** valuator for each voltage, and name them Volt1SP, Volt2SP, Volt3SP, and Volt4SP. For the voltage limits in this example use 1900mv and 2100mv for **Volt1**, 1400mv and 1600mV for **Volt2**, 900mV and 1100mV for **Volt3**, and 400mV and 600mV for **Volt4**.
To finish the transfer function for the **PowerGood** LED, use a **Priority Encoder** type transfer function, which acts like a series of “if-then-elseif” statements, the first condition to evaluate to true sets the value of the output.

Then create an “if” for each voltage checking whether the **SetPointRegion** for that voltage indicates the voltage is within the two thresholds. The “else if (I)” is the same as a final “else” condition in traditional logic blocks.
Now that we the system and its behavior has been defined, save it and move to the behavioral simulator to check the logic. As shown, when each voltage is within the specified limits, the **PowerGood** LED is set to on.

The one thing omitted from the start of this design example was the I²C slave interface. This is the easiest part of all. To add a register-based protocol I²C slave interface, one that operates as an asynchronous task to the main control application (the voltages-setpoints-LED application), all that is necessary is to drag the “**Interface**” icon onto the screen, select “**I²C Slave**” from the catalog and define the address desired.
To define the custom I\textsuperscript{2}C slave, now choose “Build” and select the PSoC device to be used (only ones with enough pins or hardware resources appear for selection). In the case of this example, choose the CY8C21234, a 16-pin SOIC device.

PSoc Express can automatically assign the pins or the user can. As shown in the figure below, put the four voltages and the PowerGood pins all on the same side. There is only one possible I\textsuperscript{2}C bus connection on this device, so those pins cannot be reassigned.
PSOC Express now generates all the firmware for the CY8C21234 device and when complete presents the user a design synopsis, including an example application schematic and bill-of-materials (in this case, this is only one LED and bias resistor) and a customized datasheet that includes the I²C register map and protocol instructions for the master.