FreeRTOS!

Welcome back to Cypress Academy. This is PSoC 6 101. In this video I will show you how to run FreeRTOS on a PSoC 6.

So, you ask yourself, why an RTOS? Well, when you have multiple cores, more memory and higher CPU frequencies, you can do a lot more. When you build devices that have multiple sensors, multiple interfaces, multiple communication blocks, etc., you need help managing all of those resources and their access to the cores. An RTOS is used just for that purpose, they're there to help you manage the complexity of your design.

FreeRTOS is one of the most commonly used open source RTOS’ in the embedded market and so I think it's a great place to start. So, let’s get right into it.

We’re going to create a new project again. This time, we’ll call it “HelloWorldFreeRTOS”.

Now we’re going to modify the build settings to automatically add FreeRTOS into our project. Right click on the project in the Workspace Explorer and select Build Settings. Pick the Peripheral Driver Library, then click FreeRTOS and choose your memory management scheme. You can read all about the memory management schemes on the FreeRTOS website. In this case we’ll just keep the default – heap\_1. Hit okay.

Then we’re going to add the digital output pin component just like we did in the previous project. We'll call it RED again, just like we did in the previous project, and we'll turn off the hardware connection.

The we'll go to the Design Wide Resources and assign the pin P0[3].

Then I'll click Generate Application…now this is really cool because when I click Generate Application this time it's going to bring in all of the drivers that we need for the pin and all of the drivers that we need for the core and the rest of the system resource, but it's also going to bring in the FreeRTOS. So, when we look in the Generated Source after the Generate Application is done, you'll be able to see the FreeRTOS source. It also adds the FreeRTOSConfig.h template file into our project.

Let’s open the FreeRTOSConfig.h file. This is the standard template from FreeRTOS. PSoC Creator copies it into your project and gives you the ability to set those configuration settings. It also puts a warning in the top of the file that when you build it will put out a warning on the console. The warning is there just to remind you to edit that file. However, all of the defaults are pretty reasonable, so all I’m going to do is comment out that warning.

Now, let’s open the main CM4.c file. I'll create a function called LED\_TASK using the standard FreeRTOS task function prototype that requires a void pointer as an argument. I'm not going to use that argument so I'll just tell the compiler to ignore that argument by putting a void around it. Now all this task is going to do is to infinitely write the GPIO for the red pin. Instead of using the CyDelay API like last time, I'm going to use the FreeRTOS task delay function, also known as vTaskDelay, and I'll delay - just as before - for 500 milliseconds.

Now in the main function, I'll create the task by using the FreeRTOS API command called xTaskCreate. I'll call the LED\_TASK function, I'll give it a name “LED\_TASK”, I'll assign a minimum stack size of 400, I'm not passing any parameters so I'll put a NULL, and then I'll pick a task priority of 1, and I don’t need a task handle so I'll give it a 0.

Next, I need to start the FreeRTOS scheduler by calling the API vTaskStartScheduler which starts and then never returns.

Now I'll build and program this to test it to make sure it works right.

Notice that we didn’t have to include any of the FreeRTOS headers. Those are already included by default in the project.h once we changed the build settings to include the FreeRTOS.

Congratulations you’re now up and running with PSoC 6, and more specifically you are running FreeRTOS in the Cortex-M4 core.

In the next set of videos, I'll walk you through the PSoC 6 peripherals that we’re going to use for the BLE-controlled robotic arm, one step at a time though.

You can post your comments and questions in our PSoC 6 developer community or as always you are welcome to email me at alan\_hawse@cypress.com or tweet me @askioexpert. Thank you!