UART

Welcome back to Cypress Academy, PSoC 6 101. In next few videos I will show you the basic building blocks to create our BLE controlled robotic arm. This will include a UART terminal interface, PWMs to control the servo motors, EZ-I2C dashboard interface, a digital logic-based kill switch, capacitive sensing controls and an advanced technique for RTOS debugging called Tracealyzer. As I go through these videos, I will first create a bare metal implementation, so you understand the basics for each of the peripheral functions and then we’ll integrate those functions with an RTOS into the BLE controlled robotic arm project.

Let’s start with the UART interface to a PC terminal client. The UART interface is great as a rudimentary debug interface when developing an application like this. It can also be used as a basic communications peripheral for other system ICs in your project.

First, we will add a new project to our previously used workspace. Let’s call it “BasicUART”.

As with our other projects, let’s drag and drop the UART component on to the schematic. Double click again to edit. Let’s call it “UART”. All the other settings look good, so just click OK.

Double click on the pins file under the Design Wide rResources and assign the RX and TX to P5[0] and P5[1].

On the back of the PSoC 6 BLE Pioneer board, you can actually see the silkscreen which tells you where all of the I/Os on this kit are. This is a nice quick reference for you as you build these projects.

So, everybody likes to use *printf*. How do you make printf work in PSoC 6? To do that I need to re-target *stdout* to the UART interface. Cypress has built a library that will allow you to re-target - we call that library *Retarget I/O*. To include this library into your project, open the build settings, click on PDL, scroll down and click the check box right next to *Retarget I/O*.

Now let’s generate the application. Once that’s done, you’ll see a file called “stdio\_user.h” that’s been generated in the Shared Files folder for you. This is where you configure the UART that you want to use for *stdout* and *stdin* to point to. Scroll down to the first few lines of code. We need to *#include project.h* so we can reference the appropriate UART below. Next, go to *#define IO\_STDOUT\_UART* and tell it which hardware block is being used for the output (in other words, which SCB you are using). PSoC Creator generates a macro for you that you can use so that you don't need to figure out which specific hardware block. In this case, the name is *UART\_HW*, which is your instance name of the UART appended with the *\_HW*. Now do the same for the #define IO\_STDIN\_UART. And now our retarget I/O library is setup.

We’re going to control the UART with the Cortex-M4, so let’s open up the main\_cm4.c. Start the UART using the API call. Standard IN is typically buffered, which means the characters you type go into a buffer, but you don’t know they’re there until you read. I want to turn that off for this program, so we can handle each character as it comes in. To do this, we’ll write a line of code: setvbuf( stdin, NULL, \_IONBF, 0). This means there will be no buffering on stdin.

For this basic project I just want to echo the characters that the PSoC 6 receives back to the terminal client. So, let’s create a character variable called c. Let’s show that printf works – amazingly it does – let's print out “Started UART” in this example project. Now, in our main loop, let’s get a character, see if anything was returned, and then print that character out to the screen.

And that’s it, now time to build, program and test it.

First, I’m going to open the Windows’ device manager to see which COM port the KitProg is attached to. You can see which COM port it’s attached to under the Ports tab. It will be labeled KitProg2 USB-UART.

Now open up your favorite terminal client and attach it to the correct COM port at 115200 baud 8-n-1. I know the baud rate and 8-n-1 settings because they're in the component configuration dialog we saw earlier.

Now in the terminal client whatever I type is echoed back to me on the screen…in this case, I type "PSoC 6 is awesome!", and I totally agree with that!

Alright, in the next video I'm going to show you how to take this UART starter project, move it into an RTOS and take the first steps in building the main robot arm controller project.

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