

Controlling a PSoC Stick from a PC Web Browser Using a Serial Link

Veloria Pannell, Eric Ponce

August 2024

1 Overview

This tutorial covers the basic hardware, firmware, and software required to control the PSoC Stick with a webpage through a serial link. The PSoC Creator 3.3 workspace along with the HTML, CSS, and JavaScript files can be found on the course website.

This manual and webpage branches from the “Controlling a PSoC Big Board from a PC Web Browser Using a Serial Link” manual, adapting the demo to accommodate the PSoC Stick (CY8CKIT-059 PSoC 5LP Prototyping Kit) instead of the PSoC Big Board (CY8CKIT-050 PSoC 5LP Development Kit). Working with the PSoC stick is more complicated than the PSoC big board, thus it is important to start with the big board demo prior to beginning this demo. While some information will be repeated in this manual, it will be focused on the changes relevant for configuring with the PSoC Stick. Refer to the Big Board Demo’s manual for the complete context and documentation of the Web-Serial interfacing.

Because the PSoC Stick connects to the PC with a USB cable, instead of a USB-to-RS232 Serial Kable, it uses a separate UART macro in PSoC Creator. This macro receives and transmits information differently than the default UART block, requiring us to handle communication differently with new JavaScript and Creator code.

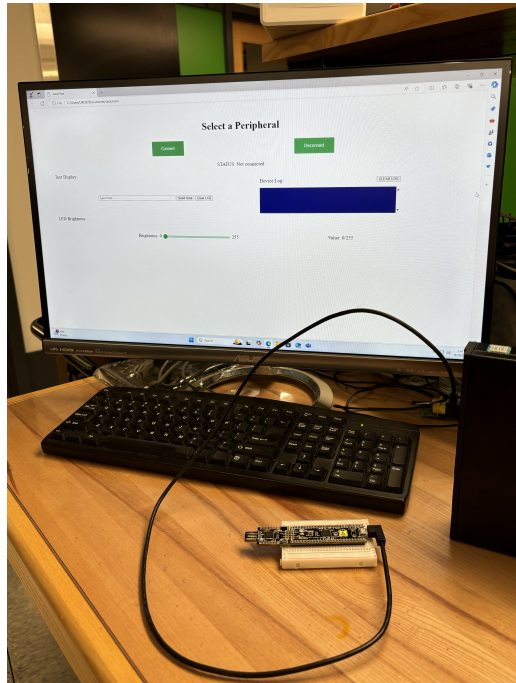


Figure 1: PC connected to PSoC Stick

1.1 How it works

The basic usage of this webpage is the same as when interfacing with the Big Board. The “Connect” and “Disconnect” buttons establish/disestablish a serial connection with the PSoC Stick, exactly the same as with the Big Board. Similarly, the slider will change the brightness of LED1 on the Stick, and text sent with the text box will be echoed by the PSoC to the Device Log.

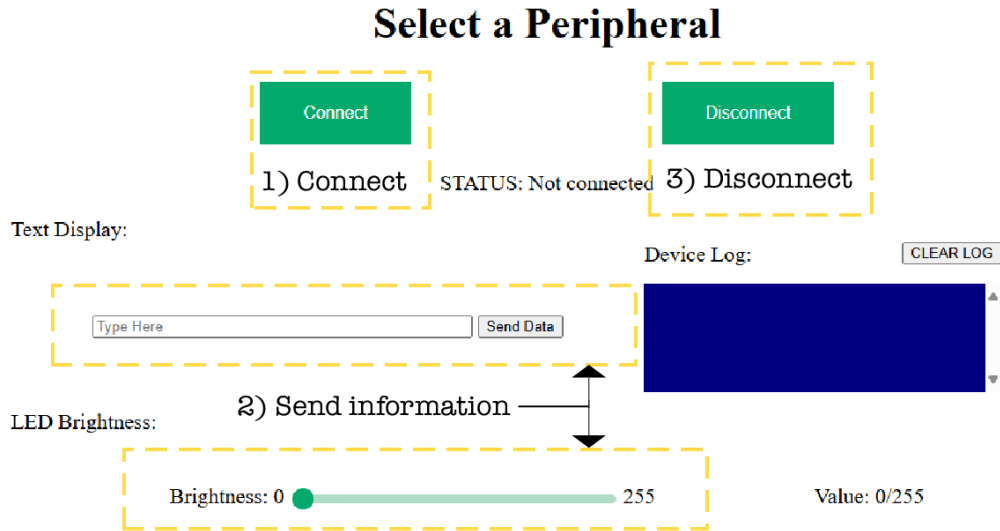


Figure 2: The webpage

2 Hardware

You will need your PSoC 5LP Stick and a mini-USB cable for both power and the serial connection. This version requires extra configuration for the USBUART Creator component; the necessary instructions are provided in the following sections. Refer to the Full Speed USB (USBFS) datasheet for anything this manual does not include (link at end of document). The Creator components and configurations are shown in more detail below (figures 3-6).

2.1 PSoC Board and Creator

After programming your stick, disconnect the KITPROG USB end and connect the mini-USB end to a USB port on a PC. LED1 is assigned to P2[1]; the USBUART is by default assigned to P15[7] and 15[6].

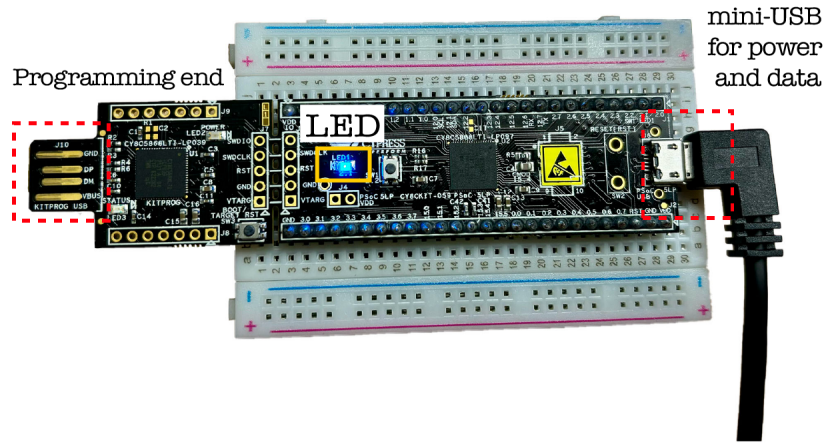


Figure 3: PSoC Stick Setup

For the Creator schematic, the required component blocks include a CDC-configured USBFS, a PWM, a clock, and an LED. The USBFS module can be found in the Cypress Component Catalog under **Communications > USB > USBUART (CDC Interface)**.

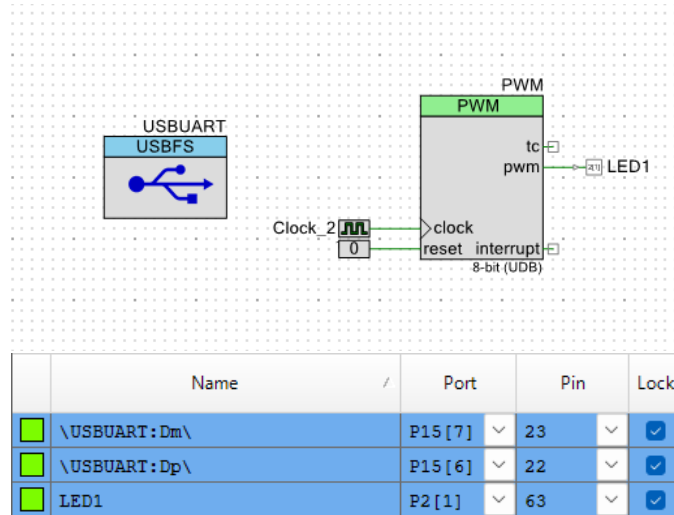


Figure 4: Example PSoC Creator Top-level Schematic and Pin Editor

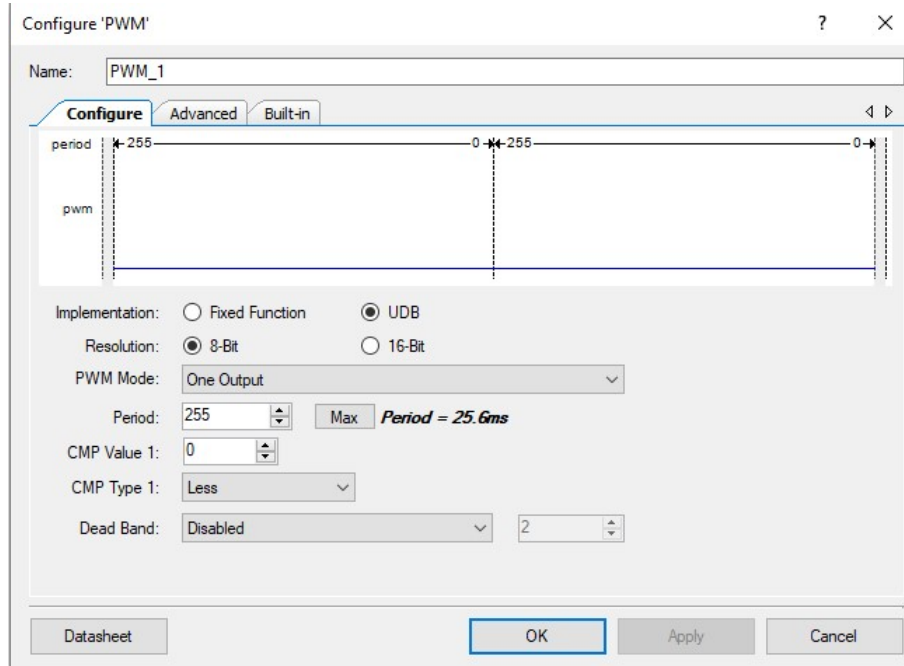


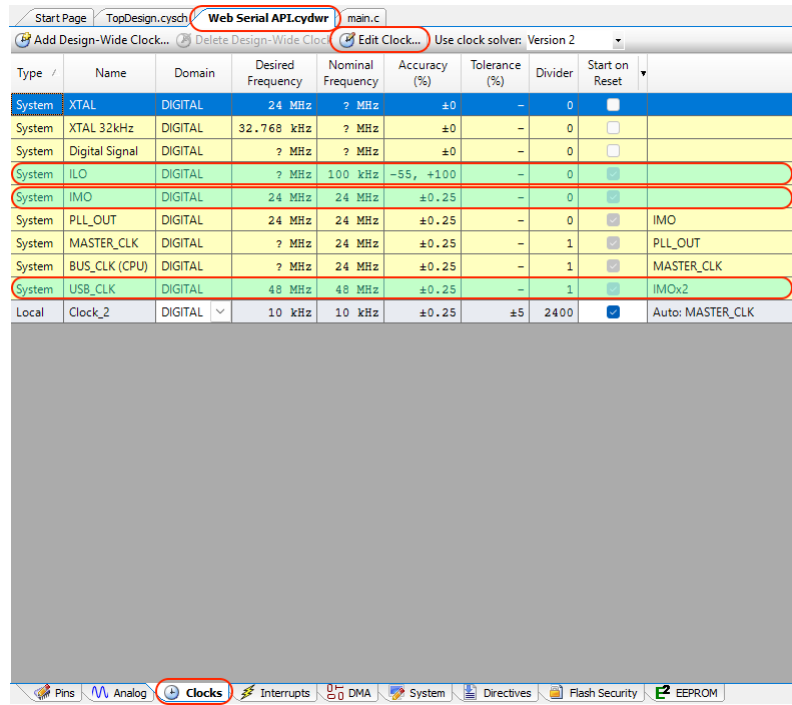
Figure 5: PWM Configuration

The PWM configuration is the same as in the other demo, but the UART is changed from the default UART block to a USBUART macro, which is a preset USBFS Component configured to implement a CDC interface (information below).

2.2 Clock Configuration for USBUART

The USBFS macro is highly configurable and can accommodate several different communication interfaces. In this case, we are using a Communications Device Class (CDC) Interface. Relevant information for CDC interfacing from the datasheet (i.e., configuration, API, variables, etc.) exists in the USBUART (CDC) section on pages 90-103.

Due to the stringent clock requirements of the USBFS component, particular settings are required for different PSoCs; this is covered in the “Quick Start” section on page 2 of the USBFS datasheet. Once all the clock requirements are met, click **Build** to generate the APIs. You can use figure 6 below to help guide you through Creator’s system clock menu and the configuration popup. The labeled steps on the figure coincide with the 3 required clock settings for PSoc 5LP devices.



In order to get to the system clock configuration menu, navigate to the .cydwr file and select the ‘Clocks’ tab at the bottom. In order to change the clock values, select one of the clocks you want to change and select the ‘Edit Clock...’ button. For any PSoC 3 or PSoC 5LP devices (i.e., the stick), there are three clocks you need to set manually:

1. **IMO:** set to 24 MHz
2. **ILO:** set to 100 kHz
3. **USB:** Enable and select IM0x2 to achieve 48 MHz

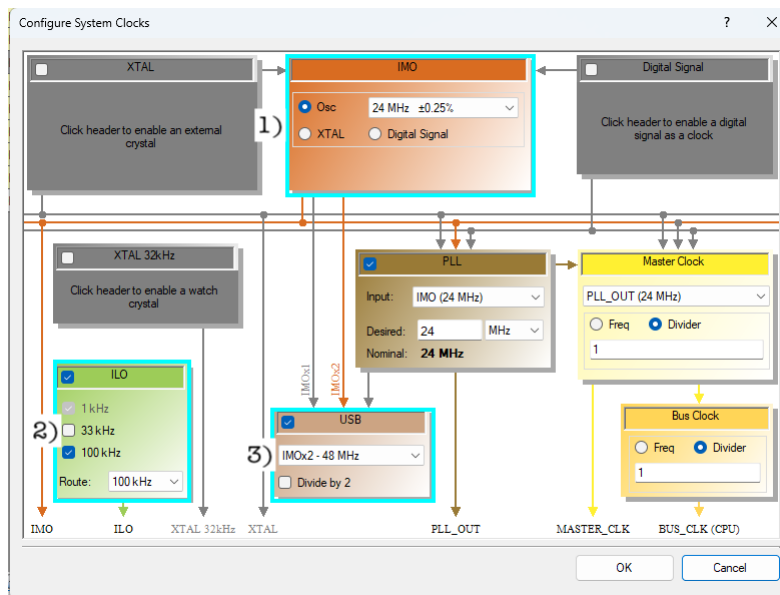


Figure 6: System Clocks Menu & Configuration Popup

3 Firmware

If you navigate to `main.c`, you will see that we are sending and receiving data using PSoC Creator's USBUART macro. Received data either alters the PWM component or echoes back to the PC. Upon device reset (either after re-programming or pressing the SW3 Button), the USBUART and PWM blocks are initialized, the LED is turned off, and the USBUART transmits “*Device Reset*,” which is printed on the Device Log on the webpage. The Stick, unlike the Big Board, fully disconnects from the webpage when it is reset, so you will receive a popup on the webpage and **need to reconnect** to the device with the ‘Connect’ button in order to continue communication with the device.

Instead of receiving information byte-by-byte like the default UART block, the USBUART macro receives information in packets of arbitrary length up to 64 bytes of data. For each packet of data, the char variable `indicator`, holds the first byte received and is checked to decide how to handle the data, detailed in Figure 7 below and throughout the rest of the manual. An indicator char of “t” is for text and “l” is for LED values (stored in the macros `TEXT` and `LED`).

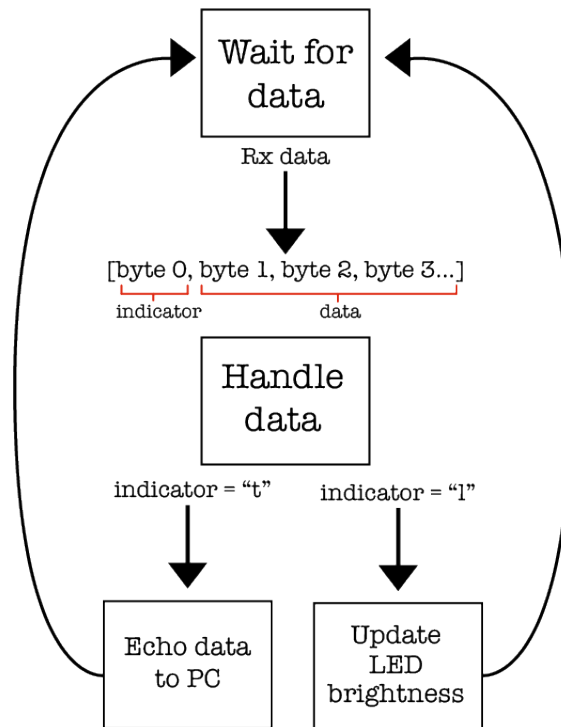


Figure 7: The main loop in the C code

Besides initialization, the main function only has 3 lines. First, we check if the USBUART is ready to send more data to the PC with the API call `USBUART_DataIsReady()`, which returns a nonzero value if the Component received data or received a zero-length packet.

```
1: if (0! = USBUART_DataIsReady())
```

To receive data, we use the API call `USBUART_GetAll()`, which gets all bytes of received data from the input buffer and places them into a specified data array. This function takes in a pointer to an array where it will store the received data and returns the number of bytes received.

```
2: count = USBUART_GetAll(pRxDataBuffer);
```

Now that we have the entire input buffer and the number of bytes we want to read, we can call the helper function `handle()` to handle the data, which takes in the pointer to our array of data and the amount of data received; the pointer is type-casted to a `char*` in order to use the `strcmp()` function to compare strings.

```
3: handle((char*)pRxDataBuffer, count);
```

Depending on the value of `indicator`, `handle()` will either update the PWM component or echo the received chars with the USBUART component as detailed below.

3.1 Sending Text

Clicking the “Send Data” button will send “t”, followed by the remainder of the user’s typed text. When received, if `indicator` is “t”, the text is returned as an echo (printed in Device Log) with the API call:

```
USBUART_PutData((pRxDataBuffer+1, count - 1));
```

The API call `USBUART_PutData()` takes in a pointer to an array and a number of bytes; it sends the inputted number of bytes from the array indicated by the pointer. In our case, since our `indicator` is not a part of the user text we do not want to echo it. To do so, we increment the pointer to the input buffer and decrement count, the number of received bytes.

3.2 Adjusting LED Brightness

When you change the value on the slider, the webpage sends “l” to the PSoC followed by the new LED value. When received, if `indicator` is “l”, the PSoC updates the PWM’s compare value with the API call:

```
PWM_WriteCompare(*(pRxDataBuffer+1));
```

Updating the PWM’s compare value alters its duty cycle; varying the duty cycle alters our perceived brightness of the LED. Here, we move our pointer to the byte after our indicator and then dereference it in order to access the value stored at the memory location the pointer points to.

4 Helpful Links

- Chrome Developer Docs – <https://developer.chrome.com/docs/capabilities/serial>
- Web Serial API Docs – <https://wicg.github.io/serial/>
- MDN Web Docs – <https://developer.mozilla.org>
- PWM datasheet – https://www.infineon.com/dgdl/Infineon-Component_PWM_V2.20-Software+Module+Datasheets-v03_03-EN.pdf?fileId=8ac78c8c7d0d8da4017d0e801c7611bd
- Full Speed USB (USBFS) datasheet – https://www.infineon.com/dgdl/Infineon-Component_USBFS_V3.2-Software%20Module%20Datasheets-v03_02-EN.pdf?fileId=8ac78c8c7d0d8da4017d0e818d2f1332