I2C Receive

My name is Alan Hawse and this is PSoC 101. Another useful hardware peripheral inside of PSoC 4 is the serial communication block, which can be used as a UART, I2C, or SPI. In this lesson I will start with PSoC 4 acting as an I2C slave and use our Windows utility called the Bridge Control Panel to play the role of master.

The Bridge Control Panel is a really useful program. All the Pioneer kits actually have two PSoC devices on them. One is the target device, and one called KitProg serves the role of programmer and debugger. Kitprog also has more functions than just program / debug. It also serves as an I2C or UART bridge between the target PSoC 4 and the computer via the USB connection. The Bridge Control Panel converts the I2C/UART protocols into USB communications. This functionality makes the Bridge Control Panel very useful for debugging embedded devices.

The Bridge Control Panel was automatically installed along with PSoC Creator so you already have it on your PC. Before we get into the I2C design we need to make sure you have the latest bridge firmware in the Kitprog. To do that launch PSoC Programmer and connect to the KitpProg target in the Port Selection panel. The PSoC Programmer software will then detect if your firmware is old. Then to make 100% sure you are up to date, Switch to the utilities tab and press the Upgrade Firmware button. It only takes a few seconds to program and verify the new image. Now we are ready to use the Control Panel. Quit PSoC Programmer.

Let’s get on with the new design now. We are going to control the brightness of an LED by updating PWM values via I2C. We will implement the slave side, receiving data from the Bridge control panel and copying it into the PWM compare register to set the duty cycle.

Make a copy of the Timer project from the previous lesson. Delete the Timer but keep the PWM. Open the customizer and change the PWM Period to 255 (which is an 8-bit number) this will allow you to only send one byte to control the brightness of the LED

Now look for I2C in the Component Catalog. You will see that there are 2 implementations. One is called simply I2C and the other is EZI2C slave. The easy version implements all of the firmware to emulate a standard EEPROM communication scheme. Add the EZI2C component to your schematic. Shorten the name and note that it has given you a default I2C address or hex 8. That address is fine – no need to change it

In the DWR file choose the appropriate pins for the I2C. The psoc101 web page shows the right pins for your device. These pins are wired directly to the kitprog bridge.

In C you need to start the component and then set up a buffer for the communications. This buffer is accessed by the interrupts that occur when the I2C transmission happen. So the buffer will get updated in the background. Define the variable and initialize the array. Then call the Ezi2cSetBuffer1 function. This function has three arguments. First it needs to know the size of the buffer. Make sure this matches the size of your array. Second it needs to know how many bytes you are allowing the master to write. In EZI2C we use a single buffer instead of individual ones for transmit and receive. To make sure the master does not accidentally overwrite private data in the buffer we use this argument to split the buffer into two pieces and protect the second part from being written. In this example we have a one byte buffer and we want the master to write into it, so we allow access to that byte. Finally you should pass in the address of the array.

As I mentioned before, the buffer gets updated by interrupts in the background, so all we need to do in the main loop is update the PWM. Of course, most of the time there will have been no change to the value and constantly updating the PWM is a bad idea so add a little code to check if the value changed before updating it. Program your kit.

Now, start the Bridge Control Panel program. The Kitprog appears in the list of connected ports. Click the kitprog in the portlist to make the connection. Now you can send data by writing a command in the control panel. The syntax for these commands is fully documented in the Help pages. For this example type a w – to indicate a write command. Then “08” (the i2c slave address specified in the component). Then ‘0’ – for the offset in the buffer. Follow that with the value for the PWM compare value in hex and close the command with the letter ‘p’ – for stop. You send the command with the “Send” button or by just hitting the return key. You will see that the LED changes brightness when you write different values to the I2C.

If the LED does not change check the output inside the Control Panel. When you send the command it echoes the command with plus signs indicating ack and minus signs indicating nak. If you get a nak then there are several possible problems. Perhaps you didn’t configure the i2c component or pins correctly, or didn’t enable the global interrupts, or didn’t start the ezi2c component or didn’t configure the buffer correctly.

If this does not work first time and you need to go back to PSoC Creator to make a change, remember to disconnect the bridge. The kitprog can only support one connection at a time and so if you forget that you will not be able to program the device from PSoC Creator. You will see an empty target selection dialog. It is not a problem though, just jump back to the Control Panel and press the Disconnect button and the target will reappear in the PSoC Creator dialog.

I am going to really test you with the next step. I want you to control two PWMs with a single write to from the Control Panel. You will need to add a new PWM, configure it, start it, and connect it to an LED pin. Make sure the LED is configured for hardware connections. Then you will need a bigger buffer and use one byte for the first PWM and the second byte for the new one. The command from the Control Panel is quite easy – just write the offset plus the two byte values between the I2C address and the final ‘p’.

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