

WES 237A: Introduction to Embedded System Design (Winter 2026)

Lab 5: Inter-Integrated Circuit (I2C) Communication

Due: 3/1/2026 11:59pm

In order to report and reflect on your WES 237A labs, please complete this Post-Lab report by the end of the weekend by submitting the following 2 parts:

- Upload your lab 5 report composed by a single PDF that includes your in-lab answers to the bolded questions in the Google Doc Lab and your Jupyter Notebook code. You could either scan your written copy, or simply type your answer in this Google Doc. **However, please make sure your responses are readable.**
- Answer two short essay-like questions on your Lab experience.

All responses should be submitted to Canvas. Please also be sure to push your code to your git repo as well.

- Connect the PMOD_AD2 peripheral to PMODA.
- Download the [iic_example.ipynb](#)
- Go through the notebook and answer the following questions. The following resources may be helpful
 - https://pynq.readthedocs.io/en/v2.6.1/pynq_libraries/pynqmb_reference.html
 - https://www.analog.com/media/en/technical-documentation/data-sheets/AD7991_7995_7999.pdf
 - https://pynq.readthedocs.io/en/v2.1/pynq_package/pynq.lib/pynq.lib.pmod.html#pynq-lib-pmod
- **What command opens a new I2C device in the MicroblazeLibrary? What are the two parameters to this command?**

```
device = liba.i2c_open(sda, scl)# TODO open a device
```

The command "i2c_open(unsigned int sda, unsigned int scl)" opens an I2C device. It takes parameters of u_ints, which would map to the pins to be dedicated to SCL(Clock) and SDA(Data).

- **What does 0x28 refer to in the following line?**
 - `device.write(0x28, buf, 1)`

The first parameter 0x28 is the slave address, which tells the I2C device what the target is, buf is the buffer data address, and 1 is the size of the data buffer.

- **Why do we write and then read when using the Microblaze Library, compared to just reading in the PMOD Library?**

The PMOD library handles the write to the Configuration Register to handle the channels, and the read is done after. This is all wrapped up in the PMOD library. The Microblaze, however, the user has to specify the write; without the write, the ADC will not know which pins to look at.

- **What does this code snippet mean?** `return ((buf[0] & 0x0F) << 8) | buf[1]`

This handles the 12-bit conversion of the AD7991. The first byte MSB contains 4 bits of status and 4 bits of DATA. The & 0x0F masks out status bits and leaves just the top 4 bits of the set of 12.

- **What is the difference between writing to the device when using the Microblaze Library and directly on the Microblaze?**

The library creates the device as an object, and the user references it as a name. You have to call the device by address and use it every time.