

WES 237A: Introduction to Embedded System Design (Winter 2024)

Lab 5: Inter-Integrated Circuit (I2C) Communication

Due: 3/3/2024 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?**

```
<i2c_device>.i2c_open(sda, scl)
class MicroblazeFunction_i2c_open(MicroblazeFunction)
| MicroblazeFunction_i2c_open(stream, index, function, return_type)
|
| Open an I2C device through an IO switch
|
| Parameters
| -----
| sda : int
|     The data pin number on the IO switch
| scl : int
|     The clock pin number on the IO switch
```

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

0x28 is the address offset. This refers to the device on the i2c channel as i2c supports multiple devices per channel.

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

When using the pmod library, we read and write straight to the peripheral. Because we are using an i2c device we have to request a response from the device. This is why we write first, then read.

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

Return buf[0] bitmasked so that we only see the 4 least significant bits, then shift those bits left by 8 bytes and bitwise OR buf[1]:

E.g

buf[0] = b'00000000 10001100

buf[1] = b'00000000 11000011

return b'00001100 11000011 = 0x0CC3

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

When using the microblaze library we are calling a preset library to write to a device on the i2c .
When writing directly to the microblaze we are writing to the peripheral itself.