```
Todo (descending order of priority)
--------------

IN PROG Make controller state machine
-   Add occasional in/out byte count monitoring messages

Add converter interface module for PMOD-DA2
-   Determine if anything architectural needs changing
-   Base module should handle adjustable bit rates and provide correct output format

Improve test framework; use to find and fix more bugs
-   Write Python test jigs for: converter board, PMOD-DA2 (later: all converter modules)
-   Write proper bit matching, format, and timing into MyHDL test framework and make unit tests
-   Design complete test case for initial PMOD-DA2 usage
-   Exercise serializer/deserializer issues
-   Use real audio signals and register programming sequences; run longer; use multiple channels simultaneously

Update FX2 firmware to use all endpoints
-   Determine proper algorithm for using only 3 flags with 4 endpoints
-   Come up with a test for the FX2 firmware using simple software and FPGA firmware (echo EP2->EP6, EP4->EP8,
measure accuracy/bitrate?)

Test simple sound card using PMOD-DA2
-   Write simple program to stream setup commands followed by audio data
-   Use logic analyzer to debug


Make register control software
-   Can be anything, but preferably a daemon that talks to a simple Python UI
-   When a setting changes, what changes are needed in the firmware in addition to writing converter registers?
    (Implement constraints)

Make converter interface modules
-   Parameterized instantiation of DAC/ADC modules using specified types
-   Look into merging control specifications so a single module handles all converters (extend differently for
DAC, ADC)

Begin documenting this architecture

Add ability for memory arbitrator to reinitialize reads/writes at the end of a cell RAM row
-   Keep track of current cell RAM address; when it reaches a multiple of 128, update the start
    address and return to initializing cycle state

Add configurable behavior to cell RAM

Make memory arbitrator capable of handling reads/writes down to 1 byte[?]
-   Properly handle upper/lower byte issues
-   Reduce min_chunk to 1 and adjust timings for correctness
-   Is this really necessary?  (If you stop playing audio, do you care if it stops 1 sample short?)
    Yes, but defer to later

Allow direction and channels to be byte values deserialized like AOVF, etc.
    (or just replace by single "port type" value)

Make simple Linux sound card driver framework
-   Alternatively, make a simple program for playing audio files that writes to EP2
-   Figure out API for configuration settings, both in SPI

Make firmware test mode that has DAC data buses looped back to ADC data buses


Completed things
----------------
DONE Add SPI interface that reads/writes configuration memory
DONE Assign I/O module registers; finish documenting registers
DONE Add deserializer for dir/chan, aovf; serializer for hwcon, chipselect to toplevel
DONE Modify FX2 interface to work with controller
DONE Document all settings/adjustments that can be made and sent over EP4 or internally performed
DONE Update fake generic DAC and ADC modules to actually consume / produce data
```