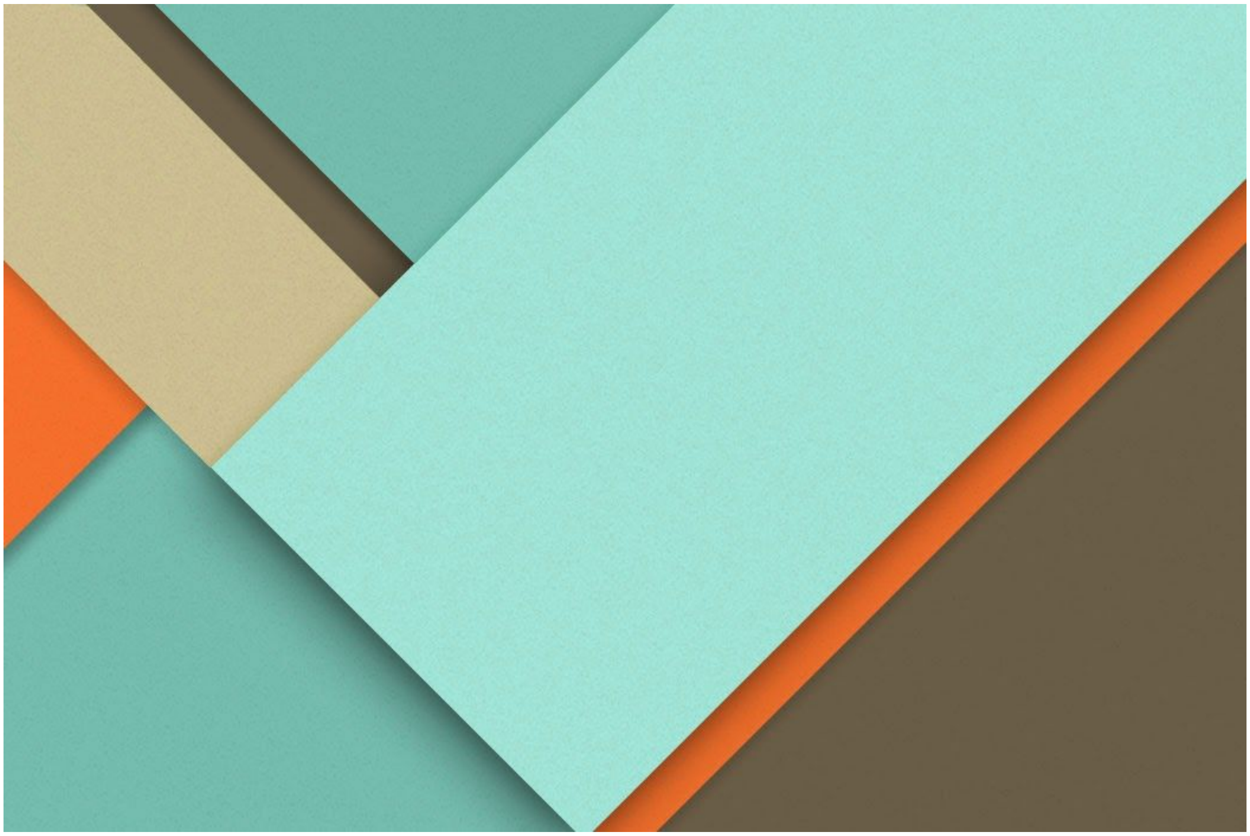

SPI EEPROM Read vs. Frequency Test

03.01.2020

Bob Honea

Overview

A SPI interface connected EEPROM must be tested across a range of interface parameters to verify function within system requirements. A Python script with supporting modules is developed to sweep testing across combinations of interface parameters for multiple target device types.

Goals

1. Support testing on present system configuration and target EEPROM spec.
2. Modular design, maintainable, upgradeable.

Specifications

Three EEPROM devices are provided as targets:

1. Microchip xxxxxxx
2. Micron 3.3V yyyyyyy
3. Micron 1.8V zzzzzzz : Vdd @ 1.7V, and 1.8V
4. Promira Serial Platform SPI/I2C/GPIO Adapter
5. Python language to operate within the Lab's Test Harness
6. Use SPI Single Data Wire and Dual Data Wire modes.

Milestones

I. Demonstrate SPI Communications

II. Demonstrate Multi-Configuration Characterization Test Sweeps

III. Demonstrate that the SPI Proxy Device meets, or fails specs.

Design Broad Outline

The test program is composed in these key modules:

I. Frequency Sweeping SPI EEPROM Read Test Driver : testspidut.py

A. Configuration management provides a readable means of configuration parameter specification, and an accessible table of supported configurations to enable characterization sweeps.

B. Test command and data read accuracy in a sweep through all supported interface and device configurations.

C. Display test results for all configurations.

II. EEPROM Command Interface : eeprom.py

A. A minimum set of commands are implemented to configure the EEPROM and SPI interface for test operation.

B. The command set varies between devices of different manufacture, and is taken into account.

III. Multimode SPI Master Transaction API : spi_io.py

A. The SPI_IO module employs Promira Active User API to manage SPI transactions through the Promira Serial Platform SPI Host Adapter.

B. Promira Serial Platform does not provide error-free operation. Error rate is at least 1 in every 10K transactions, during sustained communication. These relatively rare failures are recognized with “Promira Error”-Exceptions and fault tolerant responsive exception handling.

C. The core SPI transaction processor design prioritizes readability and maintainability. Table driven EEPROM SPI command-transaction specification keeps the transaction processor simple, maintainable, and extendable.

Supporting Features

The test program is supported by these additional modules:

IV. Test Result Reporting : xxxx_histogram.py

A. Testing is basically composed in multiple sessions of reading data from the target EEPROM with a mix of read command types. Configuration sweeps, at present are limited to slewing through a range of SPI clock frequencies.

B. Results reported for each configuration include:

1. Total Read Command Pass/Fail Counts
2. Histogram of Error Counts for each SPI clock frequency
3. Count of failures where all bytes read were a single value.
4. Report of each Promira Adapter fault exceptions.

V. EEPROM Command Table Implementation : cmd_protocol.py

A. The SPI Transaction Processor executes according to a Command Specification Tuple which enumerates each phase of a transaction's protocol, and includes operational details for each phase. The Command Specification Tuple and any relevant data origin/destination and quantity are sufficient for execution of a command.

B. The tuple is built from specifications found in the target device's datasheet. The data can be manually transcribed into structures within the command protocol module.

C. The program xxxxxxx.py is able to reduce time and errors by reducing manual operations to extracting the target device command specifications to a spreadsheet, followed by editing for consistency. The program draws the detail from the spreadsheet and generates the python Command Specification Tuples.

VI. Target EEPROM Configuration Management : spi_cfg_mgr.py

A. A minimum set of commands are implemented to configure the EEPROM and SPI interface for test operation.

1. SPI Clock Frequency
2. Target Device Logic Level and Supply
3. SPI Clock Polarity
4. SPI Clock Level
5. SPI Bit Order
6. EEPROM Base Address

B. Parameters 3 through 6 may not be changed by SPI EEPROM commands. Synchronizing changes of these device and system parameters can be done with manual edits to the configuration spec data structure.

VII. EEPROM Read/Write : testspidut.py, eeprom.py, eeprom_map.py

A. The primary function for testing is the data Read function.

B. The secondary function is to pattern-write the target device in a controlled, and convenient way, requiring a management layer for EEPROM erasure and writing.

C. EEPROM writing management is implemented with erase-before write logic.

D. Writing is limited to data aligned on 256 byte page address boundaries.

E. The writability status of the EEPROM is maintained at the Block, Sector, and Page granularities.

VIII. EEPROM Recognition, API Configuration: eeprom.py

A. The read JEDEC ID command is standard on SPI EEPROM command sets, and is used to ID the target device, and select its host-side configuration data. This recognition allows the program to self-configure whenever the JEDEC ID is accessible. In cases where the JEDEC ID is obscured, the device can be manually identified in the Configuration Parameters Table.

IX. Miscellaneous Support Features: test_util.py

A. Display and Debug Detail logging: Results Display and Debug Information can be logged to the screen or a file, or dropped based on a set of control functions.

B. Fatal errors optionally dump the most recent Detail and Debug information to screen and/or log file.

C. Functional and Fixed-Seeded Random data patterns are available to pre-format the EEPROM, and to compare read results.

D. Annotated Data Display provides a record of data received. Annotation of erroneous received bytes aids failure analysis.