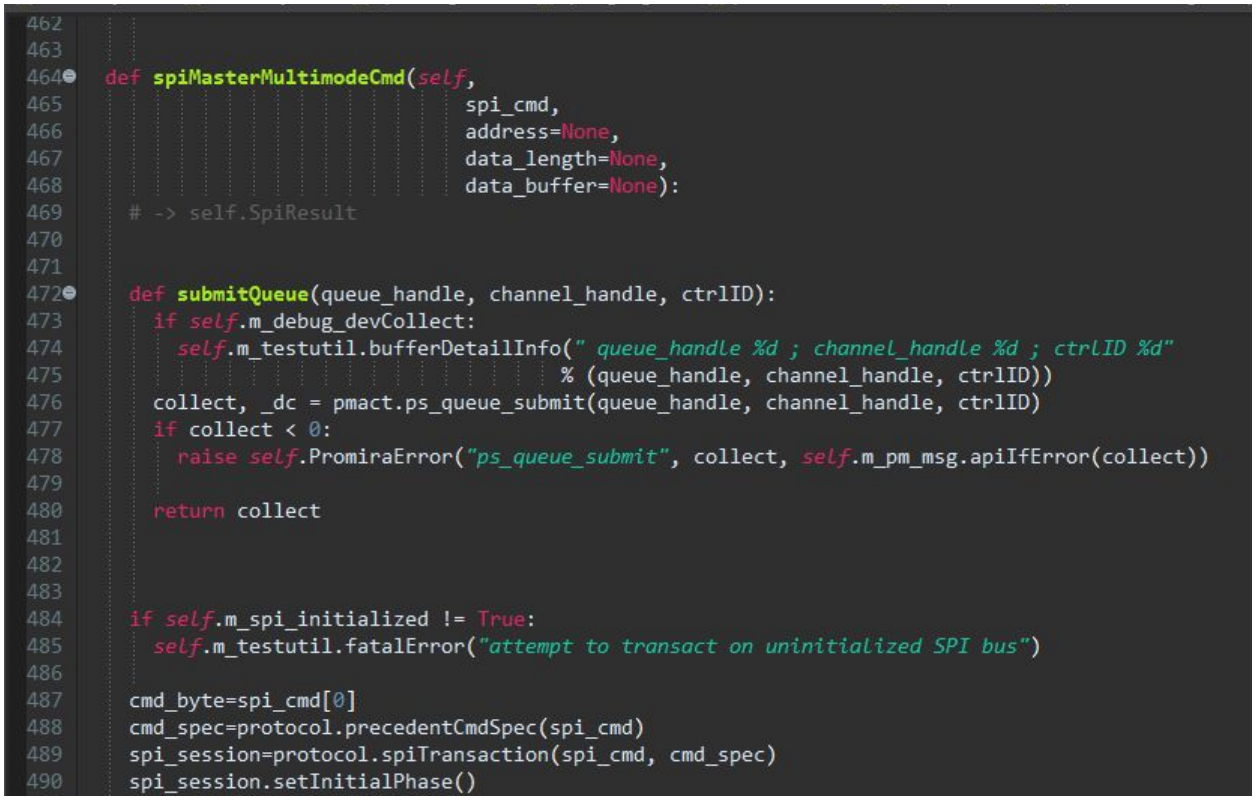

```
462
463
464   def spiMasterMultimodeCmd(self,
465                             spi_cmd,
466                             address=None,
467                             data_length=None,
468                             data_buffer=None):
469     # -> self.SpiResult
470
471
472     def submitQueue(queue_handle, channel_handle, ctrlID):
473       if self.m_debug_devCollect:
474         self.m_testutil.bufferDetailInfo(" queue_handle %d ; channel_handle %d ; ctrlID %d"
475                                          % (queue_handle, channel_handle, ctrlID))
476       collect, _dc = pmact.ps_queue_submit(queue_handle, channel_handle, ctrlID)
477       if collect < 0:
478         raise self.PromiraError("ps_queue_submit", collect, self.m_pm_msg.apiIfError(collect))
479
480       return collect
481
482
483
484     if self.m_spi_initialized != True:
485       self.m_testutil.fatalError("attempt to transact on uninitialized SPI bus")
486
487     cmd_byte=spi_cmd[0]
488     cmd_spec=protocol.precedentCmdSpec(spi_cmd)
489     spi_session=protocol.spiTransaction(spi_cmd, cmd_spec)
490     spi_session.setInitialPhase()
```

SPI EEPROM Read vs. Frequency Test

03.23.2020

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 SST26VF032B
2. Micron 3.3V MT25Qxxxx
3. Micron 1.8V MT25Qxxxx
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.

Completed Milestones

I. Demonstrate SPI Communications

II. Demonstrate Multi-Configuration Characterization Test Sweeps

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

Design 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.

Design Capabilities and Limitations

I. Target Device Protocol
   A. Per requirements, only Single bit/per/clock and Dual bit/per/clock SPI transactions are implemented and tested.
   B. Quad bit/per/clock mode transactions are implemented in the foundation modules.
   C. Quad bit/per/clock mode transactions are untested, and therefore, not reliable until tested and debugged.
   D. Multimode transactions are fully implemented, per the specification that Dual bit/per/clock mode be implemented for a Single/Single/Dual mode read command. I.e. Single Mode Cmd Phase, Single Mode Address Phase, Dual Mode Data Phase.

II. Promira SPI Parameters
   A. The maximum frequency for the Promira Spi Master transaction processing is between 31 kHz and 80 MHz.
   B. Testing has exercised the device up to 60 MHz.
   C. In direct/no-proxy SPI EEPROM testing with Promira and the target DUT board, successful transactions have occurred reliable up to and beyond 50MHz.

III. Device Power
   A. Power is set statically on the Bench Power Supply. It is not currently a proven software configurable parameter.
   B. Promira provides 3.3V, however it cannot provide BOTH 3.3v, and 1.8v for configurations where the DUT and the SPI EEPROM require different supply voltages.

IV. Promira Device Errors
   A. The Promira SPI APIs sometimes return error codes, which disrupts processing of a particular SPI EEPROM command.
   B. Exception processing has been implemented to abort the command, and restart the current sub-test.
   C. It is proven that these errors are unpredictable, though trending upward with intensity of usage.
   D. The error statistics for Promira's errors are kept with the test statistics display for transparency, and as an aid to communicating with Totalphase (Promira's Mfgr.) about this problem.

Supporting Features

The test program is supported by these additional modules:

IV. Test Result Reporting : err_fault_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. Classified data display distinguishes generic error filled pages from completely single-valued pages (all 0's or FF's). Classified error statistics Annotation classifies erroneous received bytes aids failure analysis.

E. Display of Promira Spi Adapter errors provides transparency of the devices limitations, as it may not have been intended for our high bandwidth use-case.

SPI Proxy Test Python Modules		
Name	**Description**	**Comments**
TestSPIDut	Test Runner	Class implements 'RunTest' method for harness. Impements high-level control of multi frequency (and other multivalued-parameter) SPI Proxy Test.
spi_io	SPI-Promira interface	Transaction Processor
spi_config_mgr	Generates all the tested configurations.	Table based, with code to generate all combinations of test parameters.
promactive_msg	Translates Promira error codes to text descriptions.	
err_fault_histogram	Test progress/statistics display.	Implements a histogram, and special error type histories.
eeprom	SPI-EEPROM Command Interface.	Single module for Micron and Microchip devices. (alternative per-chip modules in progress)
eeprom_map	A per page/sector/block erase/write status class.	Facilitates efficient erase/read/write during pattern writing, and other read/write activities.
eeprom_devices	Configuration tables for various target EEPROM devices.	
cmd_protocol	Transaction Descriptor Tables, and transaction descriptor class.	The 'spi_transaction' object is the template for the spiMasterMultimodeCmd transaction processor.
