Spansion Flash Memory Test Cases List
S25HL256T

Table of Contents

1 References.....5

2 Introduction.....5
3 Test Sequences.....6
3.1 Power Up.....6
3.1.1 Series 1. Power Up Negative Test.....6
a. POWERUP_1_a.....6
3.1.2 Series 2. Power Up Positive Test.....6
a. POWERUP_2_a.....6
3.2 Write Enable/Write Disable.....7
3.2.1 Series 3. Positive Test.....7
a. WREN_3_a.....7
b. WRDI_3_b 7
3.3 Initial configuration (TBPARM) 7
3.3.1 Series 4. Positive Test 7
a. TBPARM_4_a.....7
b. TBPARM_4_b.....8
3.4 Block Protection Tests 8
3.4.1 Series 5. Positive Test 8
a. PROT_5_a 8
3.5 Initial configuration (TBPROT) 10
3.5.1 Series 6. TBPROT Tests.....10
a. TBPROT_6_a.....10
3.6 Read 10
3.6.1 Series 7. Positive Tests.....10
a. READ_7_a.....10
b. READ_7_b.....11
c. READ_7_c.....11
d. READ_7_d.....12
e. READ_7_e.....13
3.7 Read Registers.....13
3.7.1 Series 8. Positive Tests.....13
a. READ_SR1_8_a.....13
3.8 Read OTP.....16
3.8.1 Series 9. Positive Tests.....16
a. READ_OTP_9_a.....16
3.9 Read ID.....16
3.9.1 Series 10. Positive Tests.....16
a. READ_JID_10_a.....16
b. READ_JQID_10_b.....17
c. READ_SFDP_and_UID_10_c.....17
3.10 WRAR TEST SR1V QPI.....17
3.10.1 Series 11. Positive Tests 17
a. WRAR_test_SR1V[4]_QPI.....17
3.11 Enter and Exit 4 Byte Address Mode.....18
3.11.1 Series 12. Positive Tests 18
a. BAM4_BAX4_12_a 18
3.12 Data Learning Pattern.....18
3.12.1 Series 13. Positive Tests 18
a. DLP_13_a 18
.....19
3.13 Software Reset.....19
3.13.1 Series 14. Software Reset Tests 19
a. RST_14_a 19

b. RST_14_b 19
3.14 OTP Program – Positive Tests.....19
3.14.1 Series 15. Positive Tests.....19
a. OTPPROG_15_a.....19
b. OTPPROG_15_b.....20
3.15 OTP Program – Negative Tests.....20
3.15.1 Series 16. Negative Tests.....20
a. OTPPROG_16_a.....20
b. OTPPROG_16_b.....21
3.16 Page Programming (3 bytes address).....21
3.16.1 Series 17. Positive Tests.....21
a. PGPROG_17_a.....21
c. PGPROG_16_c.....22
3.17 Program Suspend.....23
3.17.1 Series 18. Positive Test 23
a. PGSUSP_18_a 23
b. PGSUSP_18_b 23
3.18 Program Password Register.....23
3.18.1 Series 19. Positive Test 23
a. PGPASSWORD_19_a 23
3.19 PPB Bit Programming.....24
3.19.1 Series 20. PPB Bits Tests 24
a. PGPPB_20_a 24
3.20 PPB Erase.....24
3.20.1 Series 21. PPB Bits Tests 24
a. PPBERS_21_a 24
c. PPBERS_21_c 25
3.21 DYB Bit Programming.....25
3.21.1 Series 22. DYB Bit Programming.....25
a. DYBPG_22_a 25
3.22 Program ASP Register.....26
3.22.1 Series 23. ASP Register Tests.....26
a. ASPPG_23_a 26
3.23 Sector Erase.....26
3.23.1 Series 24. Positive Tests 26
a. SECERS_24_a 26
3.23.2 Series 25. Negative Tests 27
a. SECERS_25_a 27
3.24 Erase Suspend.....27
3.24.1 Series 26. Positive Tests 27
a. ERSSUSP_26_a 27
3.25 Program During Erase Suspend.....28
3.25.1 Series 27. Positive Tests 28
a. ERSSUSPPG_27_a 28
3.26 Software Reset.....28
3.26.1 Series 28 Positive Tests 28
a. RESET_28_a 28
b. RESET_28_b 28
c. RESET_28_c 29
3.27 Bulk Erase.....29
3.27.1 Series 29. Negative Tests 29
a. BULKERS_29_a 29

3.27.2 Series 30. Positive Tests 29
a. BULKERS_30_a 29
3.28 Configuration Register Bits For Data Protection.....30
3.28.1 Series 30. WRR Command Tests 30
b. FREEZE_30_b.....30
d. LOCK_30_d.....31
3.29 Persistent Protection Mode.....31
3.29.1 Series 31. Positive Tests 31
a. PSTPROT_31_a.....31
3.30 Deep Power Down Mode.....31
3.30.1 Series 32. Positive Test 31
a. DPDMODE_32_a.....31
3.31 DICASO.....32
3.31.1 Series 33. Positive Tests 32
a. DICASO_33_a.....32
3.32 Password Protection Mode.....32
3.32.1 Series 34. Positive Tests 32
a. PSWPROT_34_a.....32
b. PSWPROT_34_b.....32
c. PSWPROT_34_c.....32
3.32.2 Series 35. Negative Tests 33
a. PSWPROT_35_a.....33
b. PSWPROT_35_b.....33
a. SECERASE_CNT_36_a.....33
3.34 Jedec Reset.....34
a. JEDEC_RST _37_a.....34
a. Autoboot_write _38_a.....34
3.36.1 Series 39. PPB Bits Tests 34
a. PGPPB_QPI_39_a 34
3.37 PPB Erase, QPI.....35
3.37.1 Series 40. PPB Bits Tests, QPI 35
a. PPBERS_QPI_40_a 35
c. PPBERS_QPI_40_c 35
3.38.1 Series 41 Positive Tests 36
a. PSWPROT_QPI_41_a.....36
b. PSWPROT_QPI_41_b.....36
c. PSWPROT_QPI_41_c.....36
3.38.2 Series 42. Negative Tests , QPI.....37
a. PSWPROT_QPI_42_a.....37
b. PSWPROT_QPI_42_b.....37

1 References

[1] 3546270-002-12345.pdf rev I Revised May03, 2018
[2] spansion_flash_testbench_struct.pdf

2 Introduction

This document defines the test cases that will be used to verify the functional behavior and timing of the Flash Memory S25HL256T VITAL model.
Functional behavior and timing is described in the appropriate S25HL256T data sheets.
Test cases are grouped in several sequences:

- Power-up
- Write Enable/Write Disable
- Initial Configuration (TBPARM)
- Block Protection/ Sector Protection
- Initial Configuration (TBPROT)
- Read Flash Memory array (Standard and Continuous)
- DDR Read Mode (Standard and Continuous)
- Read Registers
- Read OTP
- Read ID
- OTP Programming
- Page Programming
- QPI Mode operations
- Program Suspend
- Program Registers operations
- Sector Erase
- Erase suspend
- Software Reset
- Hardware Reset
- Bulk Erase
- Configuration Register bits for data protection
- Persistent Protection mode
- Password protection mode

Each sequence consists of series of Positive and Negative tests
Each test case has the code name complying with following rule:
XXX_YYY_ZZZ
where:
XXX – Test Sequence name.
YYY – Test series number
ZZZ – Test case (a, b, c, ...)

3 Test Sequences

3.1 Power Up

3.1.1 Series 1. Power Up Negative Test

a. POWERUP_1_a

Testcase symbolic name: POWERUP_1_a
Description*:* Test if the device ignores commands written during powerup.
What is being verified:
The device ignores all instructions until a time of t_{PU} has elapsed.
Method of verification:
At simulation time 0 fs drive signals CS# low. Initiate read instruction and read SO signal to verify that read has not been accepted on the rising edge of CS#. After power up time output SO should be tristated. Warning is generated.

3.1.2 Series 2. Power Up Positive Test

a. POWERUP_2_a

Testcase symbolic name: POWERUP_2_a
Description*:* Test if the device is properly delivered with all bits set to 1 after power up and that Status and Configuration Registers contain proper values.
What is being verified:
After power-up SO signal should be Hi-Z and model should be ready to accept any instruction.
Method of verification:
After time of t_{PU} has elapsed drive signal CS# low and initiate read instruction to verify memory preloaded data. Initiate read status register instruction to verify status register bits. Initiate read configuration register instruction to verify configuration register bits.

3.2 Write Enable/Write Disable

3.2.1 Series 3. Positive Test

a. WREN_3_a

Testcase symbolic name: WREN_3_a
Description*:* Test writes to Write Enable Latch (WEL) bit.
What is being verified:
Write Enable instruction.
Method of verification:
Drive CS# signal low, initiate WREN instruction and then drive CS# signal high. Initiate read status register instruction to verify that WREN instruction sets the WEL bit.

b. WRDI_3_b

Testcase symbolic name*:* WRDI_3_b
Description*:* Test writes to Write Enable Latch (WEL) bit.
What is being verified:
Write Disable instruction.
Method of verification:
Drive CS# signal low, initiate WRDI instruction and then drive CS# signal high. Initiate read status register instruction to verify that WRDI instruction resets the WEL bit.

3.3 Initial configuration (TBPARM)

3.3.1 Series 4. Positive Test

a. TBPARM_4_a

Testcase symbolic name*:* TBPARM_4_a
Description*:* TBPARM bit positive test
What is being verified:
The desired state of TBPARM must be selected during the initial configuration of the device during system manufacture; before the first program or erase operation on the main Flash array. TBPARM must not be programmed after programming or erasing is done in the main Flash array.
Method of verification:
First, initiate WREN instruction to set WEL bit. Then, initiate WRR command with sixteen input cycles to change Configuration Register TBPARM bit. Initiate RDSR1 instruction to verify that write cycle is in progress (WIP='1'). After t_w verify that write cycle is done by checking Status Register WIP bit(WIP='0').
After write cycle is over, initiate RDCR1 command to verify that TBPARM has correct value.

b. TBPARM_4_b

Testcase symbolic name*:* TBPARM_4_b
Description*:* TBPARM bit negative test
What is being verified:
If TBPARM is already programmed to 1, an attempt to change it back to zero will fail and set the Program Error bit (P_ERR in SR1[6]).

Method of verification:
First, initiate WREN instruction to set WEL bit. Then, initiate WRR command with sixteen input cycles to change Configuration Register TBPARM bit. Initiate RDSR1 instruction to verify that operation is not accepted (WIP='0'). Verify that P_ERR bit is set. Initiate read RDCR1 instruction to verify that TBPARM bit did not change its value (TBPARM='1')

3.4 Block Protection Tests

3.4.1 Series 5. Positive Test

a. PROT_5_a

Testcase symbolic name*:* PROT_5_a
Description*:* Test Block Protection functionality for Hybrid Architecture (TBPROT = 0)
What is being verified:
The Block Protection (BP2-BP0) bits in the status register can be used to protect an address range of the main Flash array from program and erase operations. The size of the range is determined by the value of the BP bits and the upper starting point of the range is selected by the TBPROT bit of the configuration register.
Method of verification:
For each combination of BP2-BP0 bits in the status register initiate WREN instruction to set Status Register 1 WEL bit. Then with WRR command set (BP2-BP0) bits value.
Initiate again WREN instruction to set WEL bit. Then, initiate PP instruction to program to flash memory. Program location should be in protected sector. Verify that program operation failed by reading status register (WIP = '0', P_ERR='1').

b. PROT_5_b

Testcase symbolic name*:* PROT_5_b
Description*:* Test Block Protection functionality for Hybrid Architecture (TBPROT = 1)
What is being verified:
The Block Protection (BP2-BP0) bits in the status register can be used to protect an address range of the main Flash array from program and erase operations. The size of the range is determined by the value of the BP bits and the lower starting point of the range is selected by the TBPROT bit of the configuration register.
Method of verification:
For each combination of BP2-BP0 bits in the status register initiate WREN instruction to set WEL bit. Then with WRR command set (BP2-BP0) bits.
Initiate again WREN instruction to set WEL bit. Then, initiate PP instruction to program to flash memory. Program location should be in protected sector. Verify that program operation failed by reading status register (WIP = '0', P_ERR='1').

c. PROT_5_c

Testcase symbolic name*:* PROT_5_c
Description*: **Description****:* Test Block Protection functionality for Uniform Architecture (TBPROT = 0)
What is being verified:
The Block Protection (BP2-BP0) bits in the status register can be used to protect an address range of the main Flash array from program and erase operations. The size of the range is determined by the value of the BP bits and the upper starting point of the range is selected by the TBPROT bit of the configuration register.
Method of verification:
For each combination of BP2-BP0 bits in the status register initiate WREN instruction to set Status Register 1 WEL bit. Then with WRR command set (BP2-BP0) bits value.
Initiate again WREN instruction to set WEL bit. Then, initiate PP instruction to program to flash

memory. Program location should be in protected sector. Verify that program operation failed by reading status register (WIP = '0', P_ERR='1').

d. PROT_5_d

***Testcase symbolic name**:* PROT_5_d
***Description**:* Test Block Protection functionality for Uniform Architecture (TBPROT = 1)
What is being verified:
The Block Protection (BP2-BP0) bits in the status register can be used to protect an address range of the main Flash array from program and erase operations. The size of the range is determined by the value of the BP bits and the lower starting point of the range is selected by the TBPROT bit of the configuration register.
Method of verification:
For each combination of BP2-BP0 bits in the status register initiate WREN instruction to set WEL bit. Then with WRR command set (BP2-BP0) bits.
Initiate again WREN instruction to set WEL bit. Then, initiate PP instruction to program to flash memory. Program location should be in protected sector. Verify that program operation failed by reading status register (WIP = '0', P_ERR='1').

3.5 Initial configuration (TBPROT)

3.5.1 Series 6. TBPROT Tests

a. TBPROT_6_a

***Testcase symbolic name**:* TBPROT_5_a
***Description**:* TBPROT bit negative test
What is being verified:
If TBPROT is programmed to 1, an attempt to change it back to zero will fail and set the Program Error bit (P_ERR in SR1[6]).
Method of verification:
First, initiate WREN instruction to set WEL bit. Then, initiate WRR command with sixteen input cycles to change Configuration Register TBPROT bit. Verify that program operation failed by reading status register (WIP = '0', P_ERR='1').
Initiate RDCR command to verify that TBPROT bit still has previous value.

3.6 Read

3.6.1 Series 7. Positive Tests

a. READ_7_a

***Testcase symbolic name**:* READ_7_a
***Description**:* Test read (3 Bytes address and AL = '0').
What is being verified:
The device is first selected by driving the CS# chip select input pin to the logic low state.
The instruction opcode is followed by a 3-byte address (AL=0) with each bit being latched in during the rising edge of the serial SCK signal. Then the memory contents at given address is shifted out on the serial output SO pin. Each bit is shifted out at the SCK frequency during the falling edge of the serial clock SCK signal. CS# signal can be driven high after any bit of the data-out sequence is being shifted out to terminate the transaction.
Method of verification:
Initiate Read Normal mode. The specified address must be in the lower 128Mbit. Verify that correct data has been read.
Initiate Read Normal mode with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). The specified address must be in the lower 128Mbit. Verify that correct data has been read.
Initiate Fast Read mode. The specified address must be in the lower 128Mbit. Verify that correct data has been read.
Initiate Fast Read mode with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). The specified address must be in the lower 128Mbit. Verify that correct data has been read.
Initiate Dual I/O High Performance Read Mode. The specified address must be in the lower 128Mbit. Verify that correct data has been read.
Initiate Dual I/O High Performance Read Mode with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). The specified address must be in the lower 128Mbit. Verify that correct data has been read.
Initiate WREN instruction to set WEL bit. Then, initiate WRR command with sixteen input cycles to change Configuration Register QUAD bit. Initiate RDSR1 instruction to verify that write cycle is in progress (WIP='1'). After t_w verify that write cycle is done by checking Status Register WIP bit (WIP='0'). After write cycle is over, initiate RDCR1 command to verify that QUAD bit is set.
Initiate Quad I/O Read Mode. The specified address must be in the lower 128Mbit. Verify that correct data has been read.
Initiate Quad I/O Read Mode with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). The specified address must be in the lower 128Mbit. Verify that correct data has been read. Initiate Quad I/O High Performance Read

Mode. The specified address must be in the lower 128Mbit. Verify that correct data has been read.
Initiate Quad I/O High Performance Read Mode with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). The specified address must be in the lower 128Mbit. Verify that correct data has been read.
Initiate Quad I/O DDR Read Mode. The specified address must be in the lower 128Mbit. Verify that correct data has been read.
Initiate Quad I/O DDR Read Mode with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). The specified address must be in the lower 128Mbit. Verify that correct data has been read.

Repeat described reading section with Wrap Enable Mode activated.

b. READ_7_b

***Testcase symbolic name**:* READ_7_b
***Description**:* Test read.
What is being verified:
The device is first selected by driving the CS# chip select input pin to the logic low state.
The instruction opcode is followed by a 4-byte address (AL=1) with each bit being latched in during the rising edge of the serial SCK signal. Then the memory contents at given address is shifted out on the serial output SO pin. Each bit is shifted out at the SCK frequency during the falling edge of the serial clock SCK signal. CS# signal can be driven high after any bit of the data-out sequence is being shifted out to terminate the transaction.
Method of verification:
First, initiate WREN instruction to set WEL bit. Then, initiate WRAR command to change Configuration Register AL bit. Initiate RDAR instruction to verify that write cycle is in progress (WIP='1'). After t_w verify that write cycle is done by checking Status Register WIP bit(WIP='0'). Verify that AL bit is set.
Initiate Read Normal mode. Verify that correct data has been read.
Initiate Read Normal mode with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.).Verify that correct data has been read.
Initiate Fast Read mode. Verify that correct data has been read.
Initiate Fast Read mode with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). Verify that correct data has been read.
Initiate Dual I/O High Performance Read Mode. Verify that correct data has been read.
Initiate Dual I/O High Performance Read Mode with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). Verify that correct data has been read.
Initiate Quad I/O High Performance Read Mode. Verify that correct data has been read.
Initiate Quad I/O High Performance Read Mode with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). Verify that correct data has been read.
Initiate Quad I/O DDR Read Mode. The specified address must be in the lower 128Mbit. Verify that correct data has been read.
Initiate Quad I/O DDR Read Mode with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). The specified address must be in the lower 128Mbit. Verify that correct data has been read.

Repeat described reading section with Wrap Enable Mode activated.

c. READ_7_c

***Testcase symbolic name**:* READ_7_c
***Description**:* Test read.
What is being verified:
The device is first selected by driving the CS# chip select input pin to the logic low state.
The instruction opcode is followed by a 4-byte address with each bit being latched in during the rising edge of the serial SCK signal. Then the memory contents at given address is shifted out on

the serial output SO pin. Each bit is shifted out at the SCK frequency during the falling edge of the serial clock SCK signal. CS# signal can be driven high after any bit of the data-out sequence is being shifted out to terminate the transaction.
Method of verification:
Initiate Read Normal mode. Verify that correct data has been read.
Initiate Read Normal mode with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.).Verify that correct data has been read.
Initiate Fast Read mode. Verify that correct data has been read.
Initiate Fast Read mode with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). Verify that correct data has been read.
Initiate Dual I/O High Performance Read Mode. Verify that correct data has been read.
Initiate Dual I/O High Performance Read Mode with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). Verify that correct data has been read.
Initiate Quad I/O High Performance Read Mode. Verify that correct data has been read.
Initiate Quad I/O High Performance Read Mode with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). Verify that correct data has been read.
Initiate Quad I/O DDR Read Mode. The specified address must be in the lower 128Mbit. Verify that correct data has been read.
Initiate Quad I/O DDR Read Mode with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). The specified address must be in the lower 128Mbit. Verify that correct data has been read.

Repeat described reading section with Wrap Enable Mode activated.

d. READ_7_d

Testcase symbolic name*:* READ_7_d
Description*:* Test read in QPI Mode.
What is being verified:
The device is first selected by driving the CS# chip select input pin to the logic low state.
The instruction opcode is followed by a 4-byte address with each bit being latched in during the rising edge of the serial SCK signal. Then the memory contents at given address is shifted out on the serial output IO0-IO3 pins. Each bit is shifted out at the SCK frequency during the falling edge of the serial clock SCK signal. CS# signal can be driven high after any bit of the data-out sequence is being shifted out to terminate the transaction.
Method of verification:
Initiate Read Normal mode. Verify that correct data has been read.
Initiate Read Normal mode with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.).Verify that correct data has been read.
Initiate Fast Read mode. Verify that correct data has been read.
Initiate Fast Read mode with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). Verify that correct data has been read.
Initiate Dual I/O High Performance Read Mode. Verify that correct data has been read.
Initiate Dual I/O High Performance Read Mode with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). Verify that correct data has been read.
Initiate Quad I/O High Performance Read Mode. Verify that correct data has been read.
Initiate Quad I/O High Performance Read Mode with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). Verify that correct data has been read.
Initiate Quad I/O DDR Read Mode. The specified address must be in the lower 128Mbit. Verify that correct data has been read.
Initiate Quad I/O DDR Read Mode with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). The specified address must be in the lower 128Mbit. Verify that correct data has been read.

e. READ_7_e

***Testcase symbolic name**:* READ_7_e
***Description**:* Test Continuous read.
What is being verified: Address jumps can be done without exiting the Dual or Quad I/O High Performance Mode through the setting of the Mode bits. This added feature removes the need for the instruction sequence. If the Mode bits equal Axh, then the device remains in Dual or Quad I/O High Performance Read Mode, and the next address can be entered without requiring instruction opcode.
Read commands may eliminate the 8 bit opcode after the first DDR Read command sends a mode bit pattern of complementary first and second Nibbles, e.g. A5h, 5Ah, 0Fh, etc., that indicates the following command will also be the same DDR Read command. The first DDR Read command in a series starts with the 8 bit opcode, followed by address, followed by four cycles of mode bits, followed by a latency period. If the mode bit pattern is complementary, the next command is assumed to be an additional DDR Read command that does not provide opcode bits. That command starts with address, followed by mode bits, followed by latency.
Method of verification:
Initiate Dual I/O High Performance Read Mode instruction (3 bytes address) with data parameter A0h. Verify that correct data has been read. Initiate again Dual I/O High Performance Read Mode instruction. Verify that address is entered without requiring instruction opcode Verify that correct data has been read.
Initiate Dual I/O High Performance Read Mode instruction (4 bytes address) with data parameter A0h. Verify that correct data has been read. Initiate again Dual I/O High Performance Read Mode instruction. Verify that address is entered without requiring instruction opcode Verify that correct data has been read.
Initiate Quad I/O High Performance Read Mode instruction (3 bytes address) with data parameter A0h. Verify that correct data has been read. Initiate again Quad I/O High Performance Read Mode instruction. Verify that address is entered without requiring instruction opcode Verify that correct data has been read.
Initiate Quad I/O High Performance Read Mode instruction (4 bytes address) with data parameter A0h. Verify that correct data has been read. Initiate again Quad I/O High Performance Read Mode instruction. Verify that address is entered without requiring instruction opcode Verify that correct data has been read.
Initiate Quad I/O DDR Read Mode instruction (3 bytes address) with data parameter 5Ah. Verify that correct data has been read. Initiate again Quad I/O DDR Read Mode instruction. Verify that address is entered without requiring instruction opcode. Verify that correct data has been read.
Initiate Quad I/O DDR Read Mode instruction (4 bytes address) with data parameter 5Ah. Verify that correct data has been read. Initiate again Quad I/O DDR Read Mode instruction. Verify that address is entered without requiring instruction opcode. Verify that correct data has been read.

3.7 Read Registers

3.7.1 Series 8. Positive Tests

a. READ_SR1_8_a

***Testcase symbolic name**:* READ_SR1_8_a
***Description**:* Test read Status Register1.
What is being verified:
The Read Status Register1 (RDSR1) instruction opcode allows the Status Register1 contents to be read out of the SO serial output pin. It is possible to read the Status Register1 continuously by providing multiples of eight clock cycles. CS# signal can be driven high after any bit of the data-out sequence is being shifted out to terminate the transaction.
Method of verification:
Initiate Read Status Register Read (RDSR1) instruction. Verify that correct data has been read.
Initiate Read Status Register Read (RDSR1) with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). Verify that correct data has been read.

b. READ_SR2_8_b

***Testcase symbolic name**:* READ_SR2_8_b
***Description**:* Test read Status Register2.
What is being verified:
The Read Status Register2 (RDSR2) instruction opcode allows the Status Register2 contents to be read out of the SO serial output pin. It is possible to read the Status Register2 continuously by providing multiples of eight clock cycles. CS# signal can be driven high after any bit of the data-out sequence is being shifted out to terminate the transaction.
Method of verification:
Initiate Read Status Register Read(RDSR2) instruction. Verify that correct data has been read.
Initiate Read Status Register Read (RDSR2) with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). Verify that correct data has been read.

c. READ_CR1_8_c

***Testcase symbolic name**:* READ_CR1_8_c
***Description**:* Test read Configuration Register.
What is being verified:
The Configuration Register (RDCR1) instruction opcode allows the Configuration Register contents to be read out of the SO serial output pin. It is possible to read the Configuration Register continuously by providing multiples of eight clock cycles. CS# signal can be driven high after any bit of the data-out sequence is being shifted out to terminate the transaction.
Method of verification:
Initiate Configuration Register Read (RDCR1) instruction. Verify that correct data has been read.
Initiate Configuration Register Read (RDCR1) with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). Verify that correct data has been read.

d. READ_ASP_8_d

***Testcase symbolic name**:* READ_ASP_8_d
***Description**:* Test read ASP Register.
What is being verified:
The ASP Register (ASPRD) instruction opcode allows the ASP Register contents to be read out of the SO serial output pin. It is possible to read the ASP Register continuously by providing multiples of 16 clock cycles. CS# signal can be driven high after any bit of the data-out sequence is being shifted out to terminate the transaction.
Method of verification:
Initiate ASP Register Read (ASPRD) instruction. Verify that correct data has been read.
Initiate ASP Register Read (ASPRD) with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). Verify that correct data has been read.

e. READ_PASSREG_8_e

***Testcase symbolic name**:* READ_PASSREG_8_e
***Description**:* Test read Password Register.
What is being verified:
The Password Register (PASSRD) instruction opcode allows the Password Register contents to be read out of the SO serial output pin. It is possible to read the Password Register continuously by providing multiples of 64 clock cycles. CS# signal can be driven high after any bit of the data-out sequence is being shifted out to terminate the transaction.
Method of verification:
Initiate Password Register Read (PASSRD) instruction. Verify that correct data has been read.

Initiate Password Register Read (PASSRD) with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). Verify that correct data has been read.

f. READ_PPBLOCK_8_f

***Testcase symbolic name**:* READ_PPBLOCK_8_f
***Description**:* Test read PPB Lock Register.
What is being verified:
The PPB Lock Register (PLBRD) instruction opcode allows the PPB Lock Register contents to be read out of the SO serial output pin. It is possible to read the PPB Lock Register continuously by providing multiples of eight clock cycles. CS# signal can be driven high after any bit of the data-out sequence is being shifted out to terminate the transaction.
Method of verification:
Initiate PPB Lock Register Read (PLBRD) instruction. Verify that correct data has been read.
Initiate PPB Lock Register Read (PLBRD) with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). Verify that correct data has been read.

g. READ_ACCREG_8_g

***Testcase symbolic name**:* READ_ACCREG_8_g
***Description**:* Test read PPB and DYB Access Registers.
What is being verified:
The PPB Access Register (PPBRD) instruction opcode allows the PPB Access Register contents to be read out of the SO serial output pin. The device is first selected by driving the CS# chip select input pin to the logic low state. Then the instruction code is latched into the SI pin during the rising edge of the serial clock SCK signal. Then the 31-Bit address is shifted in. After address, the 8-bit PPB access register contents is shifted out on the serial output SO pin. Each bit is shifted out at the SCK frequency during the falling edge of the serial clock signal. It is possible to read the PPB Access Register continuously by providing multiples of eight clock cycles. CS# signal can be driven high after any bit of the data-out sequence is being shifted out to terminate the transaction.
The DYB Access Register (DYBRD) instruction opcode allows the DYB Access Register contents to be read out of the SO serial output pin. Procedure for reading this register is the same as for PPB Access Register.
Method of verification:
Initiate PPB Access Register Read (PPBRD) instruction. Verify that correct data has been read.
Initiate DYB Access Register Read (DYBRD) instruction. Verify that correct data has been read.

h. READ_ECCREG_8_h

***Testcase symbolic name**:* READ_ECCREG_8_g
***Description**:* Test read ECC Register.
What is being verified:
To read the ECC Status Register, the command is followed by the ECC unit address, the four least significant bits (LSB) of address must be set to zero. This is followed by the number of dummy cycles selected by the read latency value in CR2V[3:0]. Then the 8-bit contents of the ECC Register, for the ECC unit selected, are shifted out on SO 16 times, once for each byte in the ECC Unit. If CS# remains low the next ECC unit status is sent thru SO 16 times, once for each byte in the ECC Unit. The maximum operating clock frequency for the ECC READ command is 166 MHz.
Method of verification:
Initiate ECC Register Read ECCRD instruction. Verify that correct data has been read.
Initiate ECC Register Read ECCRD4 instruction. Verify that correct data has been read.

i. READ_DLPRD_8_i

***Testcase symbolic name**:* READ_DLPRD_8_i
***Description**:* Test read Data Learning Pattern.
What is being verified:
The instruction is shifted on SI, then the 8-bit DLP is shifted out on SO. It is possible to read the DLP continuously by providing multiples of eight clock cycles. The maximum operating clock

frequency for the DLPRD command is 166MHz.
Method of verification:
Initiate DLPRD instruction. Verify that correct data has been read.

j. READ_RDAR_8_j

Testcase symbolic name*:* READ_RDAR_8_j
Description*:* Test read VLDR Register.
What is being verified:
The Read Any Register (RDAR) command provides a way to read all device registers non-volatile and volatile. The instruction is followed by a 3 or 4 Byte address (depending on the address length configuration CR2V[7], followed by a number of latency (dummy) cycles set by CR2V[3:0]. Then the selected register
Reading undefined locations provides undefined data.
Method of verification:
Initiate RDAR instruction for all addresses in defined address range. Verify that correct data has been read.

3.8 Read OTP

3.8.1 Series 9. Positive Tests

a. READ_OTP_9_a

Testcase symbolic name*:* READ_OTP_9_a
Description*:* Test read OTP region.
What is being verified:
The Read OTP Data bytes commands reads data from the OTP region. The protocol of the Read OTP Data Bytes commands is the same as the Fast Read Data Bytes command except that it will not wrap to the starting address after the OTP address is at its maximum; instead, the data beyond the maximum OTP address will be undefined. CS# signal can be driven high after any bit of the data-out sequence is being shifted out to terminate the transaction.
Method of verification:
Initiate OTPR instruction from the OTP region 1. Verify that correct data has been read.
Initiate OTPR instruction from the OTP region 31. Verify that correct data has been read.
Initiate OTPR instruction from the address beyond the maximum OTP address. Verify that data is undefined.

3.9 Read ID

3.9.1 Series 10. Positive Tests

a. READ_JID_10_a

Testcase symbolic name*:* READ_JID_10_a
Description*:* Test Read JEDEC Standard ID instruction.
What is being verified:
The device is first selected by driving the CS# chip select input to the logic low state. After this, the RDID 8-bit instruction opcode is shifted in onto the SI serial input. After the last bit of the RDID instruction opcode is shifted into the device, a byte of manufacturer identification, two bytes of device identification, extended device identification and CFI information will be shifted sequentially out on the SO serial output. The Read Identification (RDID) instruction sequence is terminated by driving the CS# chip select input to the logic high state anytime during data output.
Method of verification:
Initiate JEDEC RDID instruction. Verify that manufacturer identification, device identification, extended device identification and CFI information are correct.

Initiate JEDEC RDID with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). Verify that correct data has been read.

b. READ_JQID_10_b

***Testcase symbolic name**:* READ_JQID_10_a
***Description**:* Test Quad Read JEDEC Standard ID instruction.
What is being verified:
The command is recognized only when the device is in QPI Mode (CR2V[6]=1). The instruction is shifted in on IO0-IO3. After the last bit of the instruction is shifted into the device, a byte of manufacturer identification, two bytes of device identification, extended device identification, and CFI information will be shifted sequentially out on IO0-IO3. As a whole this information is referred to as ID-CFI.
Method of verification:
Initiate JEDEC RDQID instruction. Verify that manufacturer identification, device identification, extended device identification and CFI information are correct.
Initiate JEDEC RDQID with clock_num aux_parameter (bus_data_read cycle drives CS# low for number of clock cycles that isn't multiple of eight.). Verify that correct data has been read.

c. READ_SFDP_and_UID_10_c

***Testcase symbolic name**:* READ_SFDP_and_UID_10_c
***Description**:* Test Read Serial Flash Discoverable Parameters and Unique ID
What is being verified:
Enter QPI mode. The SFDP bytes are always shifted out with the MSB first. If the 24-bit address is set to any other value, the selected location in the SFDP space is the starting point of the data read. This enables random access to any parameter in the SFDP space. Read Unique Id.
Exit QPI mode.The SFDP bytes are always shifted out with the MSB first. If the 24-bit address is set to any other value, the selected location in the SFDP space is the starting point of the data read. This enables random access to any parameter in the SFDP space. Read Unique Id.
Method of verification:
Initiate RSFDP and RDUID instructions.. Verify that correct data has been read.

3.10 WRAR TEST SR1V QPI

3.10.1 Series 11. Positive Tests

a. WRAR_test_SR1V[4]_QPI

***Testcase symbolic name**:* WRAR_11_a
***Description**:* TestWRARCommand
What is being verified:
Enter QPI mode, Initiate w_enable, then wrar command, for status register 1.

Method of verification:
This test Is for testing WRAR and RDAR in QPI mode.

3.11 Enter and Exit 4 Byte Address Mode

3.11.1 Series 12. Positive Tests

a. BAM4_BAX4_12_a

***Testcase symbolic name**:* BAM4_BAX4_12_a
***Description**:* Test Enter, Exit 4 Byte Address Mode Command
What is being verified:
The enter 4 Byte Address Mode (4BAM) command sets the volatile Address Length bit (CR2V[7]) to 1 to change most 3 Byte address commands to require 4 Bytes of address.
The exit4 Byte Address Mode (4BAX) command sets the volatile Address Length bit (CR2V[7]) to 0 to change most 4 Byte address commands to require 3 Bytes of address.
Method of verification:
Initiate WREN command to set WEL bit. If the user set WEL bit before BAM4 command is issued,
WEL bit will remain high after the command.
Initiate BAM4 instruction to Enter 4 Byte Addres Mode. Verify that AL bit is set to 1 by reading CR2V register.
Initiate FAST READ command and verify that address is 4 Bytes in length.
Initiate OTP READ command and verify that address is 4 Bytes in length.
Initiate WREN command to set WEL bit. If the user set WEL bit before BAM4 command is issued,
WEL bit will remain high after the command.
Initiate BAM4 instruction to Enter 3 Byte Addres Mode. Verify that AL bit is set to 0 by reading CR2V register.
Initiate FAST READ command and verify that address is 3 Bytes in length.
Initiate OTP READ command and verify that address is 3 Bytes in length.

3.12 Data Learning Pattern

3.12.1 Series 13. Positive Tests

a. DLP_13_a

***Testcase symbolic name**:* DLP_13_a
***Description**:* Programming NVDLR/VDLR plus reading VDLR
What is being verified:
Reading VDLR and reading memory data in DLP mode
Method of verification:
Enter QPI mode.Initiate WREN instruction to set WEL bit. Initiate PNVDLR instruction. Initiate read status register instruction to verify that PP cycle is in progress. After t_{pp} verify that programming is done by reading status register.
Initiate DLPRD command to verify that correct data is programmed.
Initiate WREN instruction to set WEL bit. Initiate WVDLR instruction. Initiate read status register instruction to verify that PP cycle is in progress. After t_{pp} verify that programming is done by reading status register.
Initiate Hardware Reset. Verify by DLPRD command that after Reset vdlr value becomes nvldr value.
For DDR read operations that have more than 5 dummy clocks verify that have an 8 edge Data Learning Pattern (DLP) driven by the memory, on all data outputs, in the dummy cycles immediately before the start of data.

3.13 Software Reset

3.13.1 Series 14. Software Reset Tests

a. RST_14_a

***Testcase symbolic name**:* RST_14_a
***Description**:* RSTEN + RST command set – negative test
What is being verified:
The Reset Enable Command is required immediately before a Reset command (RST). Any command other than RST following the RSTEN command will clear the reset enable condition and prevent a later RST command from being recognized.
Method of verification:
Initiate RSTEN instruction. Then initiate WREN command to clear reset enable condition. Initiate RST command and verify that command is ignored.

b. RST_14_b

***Testcase symbolic name**:* RST_14_b
***Description**:* RSTEN + RST command set – positive test
What is being verified:
The RST command immediately following a RSTEN command initiates the software reset process.
Method of verification:
Initiate RSTEN instruction. Then initiate RST command to start Software Reset procedure. A reset command is executed when CS# is brought high at the end of instruction and requires tRPH time to execute. After tRPH time is elapsed verify that the non-volatile bits in the configuration register CR1NV (TBPROT_NV, TBPARM_NV) retain their previous state. Verify that the Block Protection bits in the status register SR1V are reseted to their default state. Verify that after Reset VDLR value becomes NVDLR value.

3.14 OTP Program – Positive Tests

3.14.1 Series 15. Positive Tests

a. OTPPROG_15_a

***Testcase symbolic name**:* OTPPROG_15_a
***Description**:* Write to OTP region - Single Byte Programming.
What is being verified:
The OTP Program command programs data in the OTP region, which is in a different address space from the main array data. The protocol of the OTP Program command is the same as the Page Program command. Each region in the OTP memory space can be programmed one or more times, provided that the region is not locked. Subsequent OTP programming can be performed only on the unprogrammed bits.
Method of verification:
First, initiate WREN instruction to set WEL bit. Then, initiate OTP_PP instruction to program a single byte to OTP region. Initiate read status register instruction to verify that OTP_PP cycle is in progress. After t_{PP} verify that programming is done by reading status register.
After programming is over verify that correct data has been programmed. Lock programmed OTP region.

b. OTPPROG_15_b

***Testcase symbolic name**:* OTPPROG_15_b
***Description**:* Write to OTP region - OTP Page Programming.
What is being verified:

The OTP Program command programs data in the OTP region, which is in a different address space from the main array data. The protocol of the OTP Program command is the same as the Page Program command. Each region in the OTP space can be programmed one or more times, provided that the region is not locked. Subsequent OTP programming can be performed only on the unprogrammed bits.
Method of verification:
First, initiate WREN instruction to set WEL bit. Then, initiate OTP_PP instruction to program a 200 bytes to OTP region. Initiate read status register instruction to verify that OTP_PP cycle is in progress. After t_{PP} verify that programming is done by reading status register.
After programming is over verify that correct data has been programmed.

3.15 OTP Program – Negative Tests

3.15.1 Series 16. Negative Tests

a. OTPPROG_16_a

Testcase symbolic name*:* OTPPROG_16_a
Description*:* Programming locked region
What is being verified:
The OTP Program command programs data in the OTP region, which is in a different address space from the main array data. The protocol of the OTP Program command is the same as the Page Program command. Each region in the OTP space can be programmed one or more times, provided that the region is not locked. Subsequent OTP programming can be performed only on the unprogrammed bits.
Method of verification:
Initiate WREN instruction to set WEL bit. Then, initiate OTP_PP instruction. Program location should be in lock region. Verify that program operation failed by reading status register (WIP = '0', P_ERR='1'). Verify that location has not been programmed.

b. OTPPROG_16_b

Testcase symbolic name*:* OTPPROG_16_b
Description*:* Programming over address limit
What is being verified:
The OTP Program command programs data in the OTP region, which is in a different address space from the main array data. The protocol of the OTP Program command is the same as the Page Program command. Each region in the OTP space can be programmed one or more times, provided that the region is not locked. Subsequent OTP programming can be performed only on the unprogrammed bits.
Method of verification:
Initiate WREN instruction to set WEL bit. Then, initiate OTP_PP instruction. Program location should be outside OTP regions. OTP Program operations outside the valid OTP Range is

ignored without P_ERR set to '1'.

c. OTPPROG_16_c

***Testcase symbolic name**:* OTPPROG_16_c
***Description**:* programming 16 lowest address bytes
What is being verified:
Programming zeros at 16 lowest address bytes will fail and set P_ERR.
Programming ones at 16 lowest address bytes is ignored without P_ERR set to "1"
Method of verification:
Initiate WREN instruction to set WEL bit. Then, initiate OTP_PP instruction with data different that 16'hFF. Program location should be within the 16 lowest address bytes. Verify that operation failed with P_ERR = “1” and WIP = “1”. A Clear Status Register followed by Write Disable command must be sent to return device to standby state.
Initiate WREN instruction to set WEL bit. Then, initiate OTP_PP instruction with data equal to 16'hFF. Program location should be within the 16 lowest address bytes. Verify that operation is ignored and P_ERR = “0”.

3.16 Page Programming (3 bytes address)

3.16.1 Series 17. Positive Tests

a. PGPROG_17_a

***Testcase symbolic name**:* PGPROG_17_a
***Description**:* Write to memory array – 3 bytes address, Page Size = 256
What is being verified:
The Page Program instruction opcode allows bytes to be programmed in the memory. Before the Page Program instruction opcode can be accepted by the device internal state machine, a WREN instruction must be issued to set WEL bit. The Page Program instruction is entered by driving the CS# chip select input pin to the logic low state. The instruction opcode is followed by 3-byte address and at least one data byte on the SI serial input pin. Up to 256 data bytes can be placed into the SI serial input pin after the 3-bytes address. If the 8 least significant address bits are not all zero, all transmitted data goes beyond the end of the current page are programmed from the start address of the same page. The CS# chip select pin must be driven to the logic low state during the entire duration of the sequence. If more than 256 bytes are sent to the device, previously latched data are discarded and the last 256 data bytes are guaranteed to be programmed correctly within the same page.
Method of verification:
First, initiate WREN instruction to set WEL bit. Then, initiate PP instruction to program single byte. Initiate read status register instruction to verify that PP cycle is in progress. After tPP verify that programming is done by reading status register.
After programming is over verify that correct data has been programmed.
Initiate WREN instruction to set WEL bit. Then, initiate PP instruction to program 3 bytes. Initiate read status register instruction to verify that PP cycle is in progress. After tPP verify that programming is done by reading status register.
After programming is over verify that correct data has been programmed.
Initiate WREN instruction to set WEL bit. Then, initiate PP instruction to program more then 256 bytes. Initiate read status register instruction to verify that PP cycle is in progress. After tPP verify that programming is done by reading status register.
After programming is over verify that correct data has been programmed.

b. PGPROG_17_b

***Testcase symbolic name**:* PGPROG_17_b
***Description**:* Write to memory array – 3 bytes address, Page Size = 512
What is being verified:
The Page Program instruction opcode allows bytes to be programmed in the memory. Before the Page Program instruction opcode can be accepted by the device internal state machine, a WREN

instruction must be issued to set WEL bit. The Page Program instruction is entered by driving the CS# chip select input pin to the logic low state. The instruction opcode is followed by 3-byte address and at least one data byte on the SI serial input pin. Up to 512 data bytes can be placed into the SI serial input pin after the 3-bytes address. If the 8 least significant address bits are not all zero, all transmitted data goes beyond the end of the current page are programmed from the start address of the same page. The CS# chip select pin must be driven to the logic low state during the entire duration of the sequence. If more than 512 bytes are sent to the device, previously latched data are discarded and the last 512 data bytes are guaranteed to be programmed correctly within the same page.

Method of verification:

Initiate WREN instruction to set WEL bit. Then, initiate PP instruction to program more then 512 bytes. Initiate read status register instruction to verify that PP cycle is in progress. After tPP verify that programming is done by reading status register.

After programming is over verify that correct data has been programmed.

c. PGPROG_16_c

Testcase symbolic name*:* PGPROG_16_c

Description*:* Write to memory array – 4 bytes address

What is being verified:

Each region in the memory space can be programmed one or more times, provided that the sector is not locked. Subsequent programming can be performed only on the unprogrammed bits.

Method of verification:

First, initiate WREN instruction to set WEL bit. Then, initiate PP4 instruction to program single byte. Initiate read status register instruction to verify that PP4 cycle is in progress. After tPP verify that programming is done by reading status register.

After programming is over verify that correct data has been programmed.

Initiate WREN instruction to set WEL bit. Then, initiate PP4 instruction to program 3 bytes. Initiate read status register instruction to verify that PP4 cycle is in progress. After tPP verify that programming is done by reading status register.

After programming is over verify that correct data has been programmed.

Initiate WREN instruction to set WEL bit. Then, initiate PP4 instruction to program more then 512 bytes. Initiate read status register instruction to verify that PP4 cycle is in progress. After tPP verify that programming is done by reading status register.

After programming is over verify that correct data has been programmed.

3.17 Program Suspend

3.17.1 Series 18. Positive Test

a. PGSUSP_18_a

Testcase symbolic name*:* PGSUSP_18_a

Description*:* Test Program suspend operation – QPI Mode

What is being verified:

Program suspend instruction.

Method of verification:

First, initiate WREN instruction to set WEL bit. Then, initiate PP4 instruction to program to flash memory. Initiate read status register instruction to verify that PP4 cycle is in progress. Initiate ESP instruction, and verify that after tEPS command has been accepted by reading status registers.

For all read commands initiate read from the top of memory array. Verify that when the highest address is reached, the address counter will wrap around and roll back to 000000h, allowing the read sequence to be continued indefinitely.

Initiate QUAD_READ4 instruction from program suspended sector. Verify that output is undefined.

Initiate QUAD_HI_DDR_READ4 instruction from program suspended sector. Verify that output is undefined.

Initiate command that is not allowed during Program Suspend. Verify that command is ignored.

Initiate PGRS command and verify that programing again in-progress by reading status register.

After programming is over verify that correct data has been programmed.

b. PGSUSP_18_b

***Testcase symbolic name**:* PGSUSP_18_b
***Description**:* Test Program suspend operation – QPI Mode Disabled
What is being verified:
Program suspend instruction.
Method of verification:
First, initiate WREN instruction to set WEL bit. Then, initiate PP4 instruction to program to flash memory. Initiate read status register instruction to verify that PP4 cycle is in progress. Initiate ESP instruction, and verify that after tEPS command has been accepted by reading status registers.
For all read commands initiate read from the top of memory array. Verify that when the highest address is reached, the address counter will wrap around and roll back to 000000h, allowing the read sequence to be continued indefinitely.
For all read commands initiate read from program suspended sector. Verify that output is undefined.
Initiate command that is not allowed during Program Suspend. Verify that command is ignored.
Initiate PGRS command and verify that programing again in-progress by reading status register.
After programming is over verify that correct data has been programmed.

3.18 Program Password Register

3.18.1 Series 19. Positive Test

a. PGPASSWORD_19_a

***Test case symbolic name**:* PGPASSWORD_19_a
***Description**:* Test write to Password register.
What is being verified:
Password register write (PASSP) instruction.
Method of verification:
First, initiate WREN instruction to set WEL bit. Then, initiate Password register write (PASSP) command. Initiate read status register instruction to verify that program cycle is in progress. After t_{PP} verify that programming is done by reading status register. Initiate read Password register instruction to verify that correct data has been programmed.

3.19 PPB Bit Programming

3.19.1 Series 20. PPB Bits Tests

a. PGPPB_20_a

***Test case symbolic name**:* PGPPB_20_a
***Description**:* Test PPB Bit program.
What is being verified:
PPB bits can't be changed if PPB LOCK = '0';
Method of verification:
First, initiate WREN instruction to set WEL bit. Then, initiate PPB Bit program (PPBP) command. Verify that program operation failed by reading status register (WIP = '0', P_ERR='1').

b. PGPPB_20_b

***Test case symbolic name**:* PGPPB_20_b
***Description**:* Test PPB Bit program .
What is being verified:
PPB Access register write (PPBP) instruction.
Method of verification:
Initiate Hardware Reset to set value of PPB LOCK bit to '1'. After t_{RPH} initiate read PPB Lock register instruction to confirm that PPB Lock bit is set.
Initiate WREN instruction to set WEL bit. Then, initiate PPB Bit program (PPBP) command. Initiate read status register instruction to verify that program cycle is in progress. After t_{PP} verify that programming is done by reading status register. Initiate read PPB Bit instruction to verify that correct data has been programmed.

3.20 PPB Erase

3.20.1 Series 21. PPB Bits Tests

a. PPBERS_21_a

***Test case symbolic name**:* PPBERS_21_a
***Description**:* Test PPB Bit erase command.
What is being verified:
PPB bits can't be changed if PPB LOCK = '0';
Method of verification:
First, initiate WREN instruction to set WEL bit. Then, initiate PPB Lock register write (PLBWR) command to clear PPB LOCK. Initiate read status register instruction to verify that program cycle is in progress. After t_{PP} verify that programming is done by reading status register. Initiate read PPB Lock register instruction to verify that correct data has been programmed.
Initiate WREN instruction to set WEL bit. Then, initiate PPB Erase (PPBE) command. Initiate read status register instruction to verify that erase cycle is in progress. After t_{PPBE} verify that erasure is done by reading status register. Verify that erase operation failed by reading status register (WIP = '0', E_ERR='1').
Initiate Hardware Reset to set value of PPB LOCK bit to '1'. After t_{RPH} initiate read PPB Lock register instruction to confirm that PPB Lock bit is set.

b. PGPPB_21_b

***Test case symbolic name**:* PPBERS_21_b
***Description**:* Test PPB Bit erase command.
What is being verified:
PPB Access register write (PPBE) instruction.
Method of verification:
Initiate WREN instruction to set WEL bit. Then, initiate PPB Erase (PPBE) command. Initiate read status register instruction to verify that erase cycle is in progress. After t_{PPBE} verify that erasure is done by reading status register. Initiate read PPB Bit instruction to verify that PPB bit is erased.

c. PPBERS_21_c

***Test case symbolic name**:* PPBERS_21_c
***Description**:* Test PPB Bit erase command.
What is being verified:
PPB Erase command is disabled if PPBOTP = '0'
Method of verification:
Initiate WREN instruction to set WEL bit. Then, initiate PPB Bit program (PPBP) command. Initiate read status register instruction to verify that program cycle is in progress. After t_{PP} verify that programming is done by reading status register. Initiate read PPB Bit instruction to verify that correct data has been programmed.
Initiate WREN instruction to set WEL bit. Initiate ASPP command to change PPBOTP bit value to

'0'. Initiate read status register instruction to verify that program cycle is in progress. After t_{PP} verify that programming is done by reading status register. Initiate read ASPRD instruction to verify that correct data has been programmed.
Initiate WREN instruction to set WEL bit. Then, initiate PPB Erase (PPBE) command. Initiate read status register instruction to verify that erase command is rejected (WIP = '0', E_ERR='1').

3.21 DYB Bit Programming

3.21.1 Series 22. DYB Bit Programming

a. DYBPG_22_a

***Test case symbolic name**:* DYBPG_22_a
***Description**:* Test DYB Bit program.
What is being verified:
DYB Access register write (DYBWR) instruction.
Method of verification:
Initiate WREN instruction to set WEL bit. Then, initiate DYB Bit program (DYBWR) command. Initiate read status register instruction to verify that program cycle is in progress. After t_{PP} verify that programming is done by reading status register. Initiate read DYB Bit instruction to verify that correct data has been programmed.

3.22 Program ASP Register

3.22.1 Series 23. ASP Register Tests

a. ASPPG_23_a

***Test case symbolic name**:* ASPPG_23_a
***Description**:* Test write to ASP register – Illegal Condition
What is being verified:
ASPR[2:1] = “00” = Illegal condition, attempting to program both bits to zero results in a programming failure.
Method of verification:
First, initiate WREN instruction to set WEL bit. Then, initiate ASP register write (ASPP) command. Initiate read status register instruction to verify that program cycle is in progress. After t_{PP} verify that programming is failed (ASPR[2:1] = “11”, P_ERR='1').

b. ASPPG_23_b

***Test case symbolic name**:* ASPPG_23_b
***Description**:* Test write to ASP register
What is being verified:
ASP register program operation
Method of verification:
PPBOTP bit is OTP, it can't be programmed more than once. Initiate WREN instruction to set WEL bit. Then, initiate ASP register write (ASPP) command to change PPBOTP bit value. Verify that program operation failed by reading status register (WIP = '0', P_ERR='1'). Initiate read ASP instruction to verify that ASP register PPBOTP bit did not change its value.

3.23 Sector Erase

3.23.1 Series 24. Positive Tests

a. SECERS_24_a

***Test case symbolic name**:* SECERS_24_a
***Description**:* Test sector erase instruction (3 Bytes address and EXTADD = '0').
What is being verified:
Erasure of sector.
Method of verification:
Initiate WREN instruction to set WEL bit. Then, initiate SE instruction. Initiate read status register instruction to verify that SECTOR_ERASE cycle is in progress. After t_{SE} verify that erasing is done by reading status register.
After erasing is over, verify that selected sector was erased by reading from them. Read from any location inside erased sector should return 0xFF.

b. SECERS_24_b

***Test case symbolic name**:* SECERS_24_b
***Description**:* Test sector erase instruction (4 Bytes address).
What is being verified:
Erasure of sector.
Method of verification:
Initiate WREN instruction to set WEL bit. Then, initiate SE4 instruction. Initiate read status register instruction to verify that SECTOR_ERASE cycle is in progress. After t_{SE} verify that erasing is done by reading status register.
After erasing is over, verify that selected sector was erased by reading from them. Read from any location inside erased sector should return 0xFF.

3.23.2 Series 25. Negative Tests

a. SECERS_25_a

***Test case symbolic name**:* SECERS_25_a
***Description**:* Test sector erase instruction (3 Bytes address).
What is being verified:
Erasure of sector.
Method of verification:
First initiate WREN instruction to set WEL bit. Then with WRR command set (BP2-BP0) bits and protect several sectors.
Initiate again WREN instruction to set WEL bit. Then, initiate SE instruction to erase one sector of flash memory. Sector selected for erasure should be protected. Verify that erase operation failed by reading status register (WIP = '1', E_ERR='1').

3.24 Erase Suspend

3.24.1 Series 26. Positive Tests

a. ERSSUSP_26_a

***Test case symbolic name**:* ERSSUSP_26_a
***Description**:* Test Erase suspend operation (3 Bytes address).
What is being verified: Erase suspend instruction.
Method of verification:
First, initiate WREN instruction to set WEL bit. Then, initiate SECTOR_ERASE instruction to erase sector in flash memory. Initiate read status register instruction to verify that SECTOR_ERASE

cycle is in progress. Initiate ERSP instruction, and verify that after t_{ESL} command has been accepted by reading status register.
Initiate READ instruction from sector not selected for programming. Verify that correct data has been read.
Initiate READ instruction from sector being program-suspended. Verify that output is undefined.
Initiate FAST_READ instruction from sector not selected for programming. Verify that correct data has been read.
Initiate FAST_READ instruction from sector being program-suspended. Verify that output is undefined.
Initiate DUAL_READ instruction from sector not selected for programming. Verify that correct data has been read.
Initiate DUAL_READ instruction from sector being program-suspended. Verify that output is undefined.
Initiate READ Configuration Register instruction. Verify that correct data has been read.
Initiate READ PPB Lock Register instruction. Verify that correct data has been read.
Initiate EPR command and verify that erasing is again in progress by reading status register.
After erasing is over verify that selected sector was erased by reading from them. Read from any location inside erased sector should return 0xFF.

3.25 Program During Erase Suspend

3.25.1 Series 27. Positive Tests

a. ERSSUSPPG_27_a

***Test case symbolic name**:* ERSSUSPPG_27_a
***Description**:* Test Program during Erase suspend operation .
What is being verified:
Program instruction during erase suspend.
Method of verification:
Initiate WREN instruction to set WEL bit. Then, initiate sector erase instruction. Check WIP bit of status register to see if erase is in progress. Initiate erase suspend instruction and check if WIP bit is de-asserted. Initiate page program instruction on sector that is not suspended. Check WIP bit to see if programming has started. Initiate program suspend instruction. Initiate series of read instructions for memory array to check data. For read instructions that address the suspended sectors memory should output 'X' ('U'). Initiate write bank register instruciton and check if write was successful by reading bank register. Initiate program resume instruction and check WIP bit to see if programing is in progress. Initiate read instruction to check programmed data.Initiate erase resume program and check WIP bit to see if erase is in progress. Initiate read array instructions to check if sector is erased (all ones)

3.26 Software Reset

3.26.1 Series 28 Positive Tests

a. RESET_28_a

***Test case symbolic name**:* RESET_28_a
***Description**:* Page Program algorithm termination.
What is being verified:
RESET instruction applied during Page Program Operation terminate programming.
Method of verification:
Initiate WREN instruction to set WEL bit. Initiate PP instruction and by reading status register verify that PP cycle is in progress.
During programming operation initiate RESET instruction. Verify by reading from status register that PP cycle is terminated. Initiate Read operation from location inside selected page. Read from any location inside page should return undefined data.

b. RESET_28_b

***Test case symbolic name**:* RESET_28_b
***Description**:* Sector Erase algorithm termination.
What is being verified:
RESET instruction applied during Sector Erase terminate operation in progress.
Method of verification:
Initiate WREN instruction to set WEL bit. Initiate SECTOR_ERASE instruction and by reading status register verify that SECTOR_ERASE cycle is in progress.
During erasing operation initiate RESET instruction. Verify by reading from status register that SECTOR_ERASE cycle is terminated. Initiate Read operation from location inside selected sector. Read from any location inside sector should return undefined data.

c. RESET_28_c

***Test case symbolic name**:* RESET_28_c
***Description**:* Program Suspend algorithm termination.
What is being verified:
If a Reset is performed while program is suspended, the suspend operation will abort, and the contents of the page that was being programmed and subsequently suspended will be undefined.
Method of verification:
First, initiate WREN instruction to set WEL bit. Then, initiate PP instruction to program to flash memory. Initiate read status register instruction to verify that PP cycle is in progress.
Initiate PGSP instruction, and verify that after 10us command has been accepted by reading status register.
During program suspend initiate RESET instruction. Verify by reading from status register that PP cycle is terminated.
Initiate Read operation from location inside selected page. Read from any location inside page should return undefined data.

3.27 Bulk Erase

3.27.1 Series 29. Negative Tests

a. BULKERS_29_a

***Test case symbolic name**:* BULKERS_29_a
***Description**:* Test Bulk Erase operation
What is being verified:
If at least one of sectors are software protected, Bulk Erase Operation will be rejected.
Method of verification:
First, initiate WREN instruction to set WEL bit. Then with WRR command set some sectors in protected state.
Initiate WREN instruction to set WEL bit. Initiate BE instruction and by reading status register verify that BE cycle is not in progress.
Initiate READ from several different locations in memory. Verify that correct data has been read.

3.27.2 Series 30. Positive Tests

a. BULKERS_30_a

***Test case symbolic name**:* BULKERS_30_a

***Description**:* Test Bulk Erase operation
What is being verified:
Bulk Erase Operation.
Method of verification:
Initiate WREN instruction to set WEL bit. Initiate WRR instruction to unprotect all sectors. After t_W verify that writing is done by reading status register.
Initiate WREN instruction to set WEL bit. Initiate BE instruction and by reading status register verify that BULK_ERASE cycle is in progress. After t_{BE} verify that erasing is done by reading status register.
After erasing is over, verify that flash memory was erased by reading from them. Read from any location inside flash memory should return 0xFFFF.

3.28 Configuration Register Bits For Data Protection

3.28.1 Series 30. WRR Command Tests

a. SRWD_30_a

***Testcase symbolic name**:* SRWD_30_a
***Description**:* Test WRR when SRWD is '1'
What is being verified:
Write status register command is ignored when SRWD = '1'
Method of verification:
Initiate WREN instruction to set WEL bit. Then, initiate WRR instruction(write 1 to SRWD bit). Initiate read status register instruction to verify that WRR cycle is in progress. After tw verify that writing is done by reading status register. Initiate WREN instruction to set WEL bit. Then, initiate another WRR instruction and verify that instruction is ignored (WEL = '0') because SRWD is '1' an WP# is low ('0').

b. FREEZE_30_b

***Testcase symbolic name**:* FREEZE_30_b
***Description**:* Test FREEZE bit.
What is being verified:
BP2-0 Bits, can not be changed once FREEZE is set to 1
Method of verification:
Initiate WREN instruction to set WEL bit. Then, initiate WRR instruction(write 1 to FREEZE bit). Initiate read status register instruction to verify that WRR cycle is in progress. After tw verify that writing is done by reading configuration register. Initiate WREN instruction to set WEL bit. Then, initiate another WRR instruction and verify that BP2-0 don't change old values.

c. FREEZE_30_c

***Testcase symbolic name**:* FREEZE_30_b
***Description**:* Test FREEZE bit.

What is being verified:
OTP array can't be programmed when FREEZE is set to '1'.
Method of verification:
Initiate WREN instruction to set WEL bit. Then, initiate WRR instruction(write 1 to FREEZE bit). Initiate read status register instruction to verify that WRR cycle is in progress. After tw verify that writing is done by reading configuration register. Initiate read OTP instruction to check OTP data. Initiate WREN instruction to set WEL bit. Then, initiate OTP program instruction. Check WIP bit to see if programing has started. Initiate soft reset instruction, FREEZE bit shouldn't be reseted. Initiate read configuration register instruction to check FREEZE bit. Initate hardware reset isntruction that will reset FREEZE bit, after that initiate read configuration register instruction to check if FREEZE bit is '0'.

d. LOCK_30_d

Testcase symbolic name*:* LOCK_*3*0_d
Description*:* Test LOCK bit.
What is being verified:
TBPROT,BPNV and BP2-0, can not be changed once LOCK is set to 1
Method of verification:
Initiate WREN instruction to set WEL bit. Then, initiate WRR instruction(write 1 to LOCK bit).
Initiate read status register instruction to verify that WRR cycle is in progress. After tw verify that writing is done by reading config register.
Initiate WREN instruction to set WEL bit. Then, initiate another WRR instruction and verify that BP2-0 don't change old values.

3.29 Persistent Protection Mode

3.29.1 Series 31. Positive Tests

a. PSTPROT_31_a

Testcase symbolic name*:* PSTPROT_*3*1_a
Description*:* Persistent Protection Mode Test
What is being verified:
Selection of Protection mode. Verification of PPB Lock bit (The Persistent Protection method sets the PPB Lock Bit to '1' during Hardware Reset, so that PPB bits are unprotected by a device reset)
Method of verification:
Initiate WREN instruction to set WEL bit. Then, initiate ASPP instruction to set conditions for Persistent Protection mode(ASPR[2:1] = “10”) and by reading status register verify that program cycle is in progress. After t_{PP} verify that programming is done by reading status register.
Initiate read ASP instruction to verify that correct data has been programmed.
Drive RST# signal low. After RST# goes high device is ready to read data. Initiate read of sPPB LOCK register and verify that PPB LOCK Bit is set to '1'.

3.30 Deep Power Down Mode

3.30.1 Series 32. Positive Test

a. DPDMODE_32_a

Testcase symbolic name*:* DPDMODE_*32*_a
Description*: Deep* Power Down Mode Test
What is being verified:
Selection of *Deep* Power Down mode. Verification of Read Instructions are ignored during Power Down. Selection of Resume from *Deep* Power Down mode
Method of verification:

Initiate DPD instruction. After t_{DPD} verify that Read Instructions are ignored by reading status register (has a Hi-Z value). Initiate RES_DPD instruction. After t_{RES} verify that device is exit from DPD mode. Also, all of this is verified for QPI mode.

3.31 DICASO

3.31.1 Series 33. Positive Tests

a. DICASO_33_a

***Testcase symbolic name**:* DICASO_*33*_a
***Description**:* Test DIC ASO, CRC calculation and read of CRC register
What is being verified:
Entering DIC ASO and starting CRC calculation over a part of memory array. Read of CRC registers to verify correct calculation.
Method of verification:
Initiate Set DICASO enter instruction, set starting address for CRC calculation and set ending address for CRC calculation. Then, by reading status register verify that write cycle is in progress. After $t_{CRCSETUP}$ verify that calculation is done by reading status register. nitiate read RDAR instruction to verify that correct data has been read.

3.32 Password Protection Mode

3.32.1 Series 34. Positive Tests

a. PSWPROT_34_a

***Testcase symbolic name**:* PSWPROT_*34*_a
***Description**:* Password Protection Mode Test
What is being verified:
If the Password Mode is chosen, the password must be programmed prior to setting the Protection Mode Lock Bits.
Method of verification:
Initiate WREN instruction to set WEL bit. Then, initiate Password register write (PASSP) command. Initiate read status register instruction to verify that program cycle is in progress. After t_{PP} verify that programming is done by reading status register.
Initiate read Password register instruction to verify that correct password has been programmed.

b. PSWPROT_34_b

***Testcase symbolic name**:* PSWPROT_*34*_b
***Description**:* Password Protection Mode Test
What is being verified:
Selection of Password mode. Verification of PPB Lock bit (The Password Protection method clears the PPB Lock Bit to '0' during Hardware Reset, so that PPB bits are protected by a device reset)
Method of verification:
Initiate WREN instruction to set WEL bit. Then, initiate ASPP instruction to set conditions for Password Protection mode(ASPR[2:1] = '01') and by reading status register verify that program cycle is in progress. After t_{PP} verify that programming is done by reading status register.
Initiate read ASP instruction to verify that correct data has been programmed.
Drive RST# signal low. After RST# goes high device is ready to read data. Initiate read of PPB LOCK register and verify that PPB LOCK Bit is cleared to '0'.

c. PSWPROT_34_c

***Testcase symbolic name**:* PSWPROT_*34*_c
***Description**:* Password Protection Mode Test
What is being verified:
Password Unlock command sequence
Method of verification:
Initiate read PPB Lock register command to verify that PPB Lock bit is cleared (PPB bits are in protected state).
Initiate WREN instruction to set WEL bit. Then, initiate Password unlock (PASSU) command. Initiate read status register instruction to verify that password unlock cycle is in progress.
After t_{pp} verify that password unlock cycle is done by reading status register. Initiate read of PPB Lock register and verify that PPB Lock Bit is set to '1'.
Drive RST# signal low. After RST# goes high device is ready to read data. Initiate read of PPB Lock register and verify that PPB Lock Bit is cleared to '0'.

3.32.2 Series 35. Negative Tests

a. PSWPROT_35_a

***Testcase symbolic name**:* PSWPROT_*35*_a
***Description**:* Password Protection Mode Test
What is being verified:
Password read and program can't be done if Password protection mode is set.
Method of verification:
Initiate WREN instruction to set WEL bit. Then, initiate Password register write (PASSP) command. Initiate read status register instruction to verify that program cycle is not in progress.
Initiate read Password register instruction and verify that command is ignored.

b. PSWPROT_35_b

***Testcase symbolic name**:* PSWPROT_*35*_b
***Description**:* Password Protection Mode Test
What is being verified:
Password unlock: wrong password entered
Method of verification:
Initiate read PPB Lock register command to verify that PPB Lock bit is cleared (PPB bits are in protected state).
Initiate WREN instruction to set WEL bit. Then, initiate Password unlock (PASSU) command with incorrect password. Initiate read status register instruction to verify that password unlock cycle is in progress.
After t_{PP} verify that password unlock cycle is done by reading status register. Verify by reading status register that programming error occurred.
Initiate read of PPB Lock register and verify that PPB Lock Bit is still cleared to '0'.

3.33 Series 36. Sector Erase Count Positive test

a. SECERASE_CNT_36_a

***Testcase symbolic name**:* SECERASE_CNT_*36*_a
***Description**: Read Sector Erase Count positive tests*
What is being verified:
Read Sector Erase Count positive tests
Method of verification:
Initiate Sector Erase command and verify Sector erase counter by reading SECV registar with RDARG_4_0 command.

3.34 Jedec Reset

3.34.1 Series 37. Positive Tests

a. JEDEC_RST _37_a

Testcase symbolic name*:* JEDEC_RST_*30*_a
Description*: JEDEC reset, positive test*
What is being verified:
JEDEC reset, positive test
Method of verification:
Initiate jedec reset.

3.35 Autooboot reg write, QPI

3.35.1 Series 38. Positive Tests

a. Autoboot_write _38_a

Testcase symbolic name*:* Autoboot_write _38_a
Description*: w_autoboot, positive test*
What is being verified:
Autoboot register programing
Method of verification:
RDAR address 00000042

3.36 PPB Bit Programming, QPI

3.36.1 Series 39. PPB Bits Tests

a. PGPPB_QPI_39_a

Test case symbolic name*:* PGPPB_QPI_39_a
Description*:* Test PPB Bit program.
What is being verified:
PPB bits can't be changed if PPB LOCK = '0';
Method of verification: Enter QPI,
First, initiate WREN instruction to set WEL bit. Then, initiate PPB Bit program (PPBP) command. Verify that program operation failed by reading status register (WIP = '0', P_ERR='1').

b. PGPPB_QPI_39_b

Test case symbolic name*:* PGPPB_QPI_39_b
Description*:* Test PPB Bit program .
What is being verified:
PPB Access register write (PPBP) instruction.
Method of verification:
Initiate Hardware Reset to set value of PPB LOCK bit to '1'. After t_{RPH} initiate read PPB Lock register instruction to confirm that PPB Lock bit is set.
Initiate WREN instruction to set WEL bit. Then, initiate PPB Bit program (PPBP) command. Initiate read status register instruction to verify that program cycle is in progress. After t_{PP} verify that

programming is done by reading status register. Initiate read PPB Bit instruction to verify that correct data has been programmed.

3.37 PPB Erase, QPI

3.37.1 Series 40. PPB Bits Tests, QPI

a. PPBERS_QPI_40_a

***Test case symbolic name**:* PPBERS_QPI_40_a
***Description**:* Test PPB Bit erase command.
What is being verified:
PPB bits can't be changed if PPB LOCK = '0';
Method of verification:
First, initiate WREN instruction to set WEL bit. Then, initiate PPB Lock register write (PLBWR) command to clear PPB LOCK. Initiate read status register instruction to verify that program cycle is in progress. After t_{PP} verify that programming is done by reading status register. Initiate read PPB Lock register instruction to verify that correct data has been programmed.
Initiate WREN instruction to set WEL bit. Then, initiate PPB Erase (PPBE) command. Initiate read status register instruction to verify that erase cycle is in progress. After t_{PPBE} verify that erasure is done by reading status register. Verify that erase operation failed by reading status register (WIP = '0', E_ERR='1').
Initiate Hardware Reset to set value of PPB LOCK bit to '1'. After t_{RPH} initiate read PPB Lock register instruction to confirm that PPB Lock bit is set.

b. PGPPB_QPI_40_b

***Test case symbolic name**:* PPBERS_QPI_40_b
***Description**:* Test PPB Bit erase command.
What is being verified:
PPB Access register write (PPBE) instruction.
Method of verification:
Initiate WREN instruction to set WEL bit. Then, initiate PPB Erase (PPBE) command. Initiate read status register instruction to verify that erase cycle is in progress. After t_{PPBE} verify that erasure is done by reading status register. Initiate read PPB Bit instruction to verify that PPB bit is erased.

c. PPBERS_QPI_40_c

***Test case symbolic name**:* PPBERS_QPI_40_c
***Description**:* Test PPB Bit erase command.
What is being verified:
PPB Erase command is disabled if PPBOTP = '0'
Method of verification:
Initiate WREN instruction to set WEL bit. Then, initiate PPB Bit program (PPBP) command. Initiate read status register instruction to verify that program cycle is in progress. After t_{PP} verify that programming is done by reading status register. Initiate read PPB Bit instruction to verify that correct data has been programmed.
Initiate WREN instruction to set WEL bit. Initiate ASPP command to change PPBOTP bit value to '0'. Initiate read status register instruction to verify that program cycle is in progress. After t_{PP} verify that programming is done by reading status register. Initiate read ASPRD instruction to verify that correct data has been programmed.

Initiate WREN instruction to set WEL bit. Then, initiate PPB Erase (PPBE) command. Initiate read status register instruction to verify that erase command is rejected (WIP = '0', E_ERR='1').

3.38 Password Protection Mode, QPI

3.38.1 Series 41 Positive Tests

a. PSWPROT_QPI_41_a

***Testcase symbolic name**:* PSWPROT_*QPI_41*_a
***Description**:* Password Protection Mode Test
What is being verified:
If the Password Mode is chosen, the password must be programmed prior to setting the Protection Mode Lock Bits.
Method of verification: Enterr QPI
Initiate WREN instruction to set WEL bit. Then, initiate Password register write (PASSP) command. Initiate read status register instruction to verify that program cycle is in progress.
After t_{PP} verify that programming is done by reading status register.
Initiate read Password register instruction to verify that correct password has been programmed.

b. PSWPROT_QPI_41_b

***Testcase symbolic name**:* PSWPROT_*QPI_41*_b
***Description**:* Password Protection Mode Test
What is being verified:
Selection of Password mode. Verification of PPB Lock bit (The Password Protection method clears the PPB Lock Bit to '0' during Hardware Reset, so that PPB bits are protected by a device reset)
Method of verification:
Initiate WREN instruction to set WEL bit. Then, initiate ASPP instruction to set conditions for Password Protection mode(ASPR[2:1] = '01') and by reading status register verify that program cycle is in progress. After t_{PP} verify that programming is done by reading status register.
Initiate read ASP instruction to verify that correct data has been programmed.
Drive RST# signal low. After RST# goes high device is ready to read data. Initiate read of PPB LOCK register and verify that PPB LOCK Bit is cleared to '0'.

c. PSWPROT_QPI_41_c

***Testcase symbolic name**:* PSWPROT_*QPI_41*_c
***Description**:* Password Protection Mode Test
What is being verified:
Password Unlock command sequence
Method of verification:
Initiate read PPB Lock register command to verify that PPB Lock bit is cleared (PPB bits are in protected state).
Initiate WREN instruction to set WEL bit. Then, initiate Password unlock (PASSU) command. Initiate read status register instruction to verify that password unlock cycle is in progress.
After t_{pp} verify that password unlock cycle is done by reading status register. Initiate read of PPB Lock register and verify that PPB Lock Bit is set to '1'.
Drive RST# signal low. After RST# goes high device is ready to read data. Initiate read of PPB Lock register and verify that PPB Lock Bit is cleared to '0'.

3.38.2 Series 42. Negative Tests , QPI

a. PSWPROT_QPI_42_a

***Testcase symbolic name**:* PSWPROT_*QPI_42*_a
***Description**:* Password Protection Mode Test
What is being verified:
Password read and program can't be done if Password protection mode is set.
Method of verification:
Initiate WREN instruction to set WEL bit. Then, initiate Password register write (PASSP) command. Initiate read status register instruction to verify that program cycle is not in progress.
Initiate read Password register instruction and verify that command is ignored.

b. PSWPROT_QPI_42_b

Testcase symbolic name*:* PSWPROT_*QPI_42*_b
Description*:* Password Protection Mode Test
What is being verified:
Password unlock: wrong password entered
Method of verification:
Initiate read PPB Lock register command to verify that PPB Lock bit is cleared (PPB bits are in protected state).
Initiate WREN instruction to set WEL bit. Then, initiate Password unlock (PASSU) command with incorrect password. Initiate read status register instruction to verify that password unlock cycle is in progress.
After t_{PP} verify that password unlock cycle is done by reading status register. Verify by reading status register that programming error occurred.
Initiate read of PPB Lock register and verify that PPB Lock Bit is still cleared to '0'