AN96589

Designing with Cypress 1-Mb Quad SPI nvSRAM

Author: Suhail Zain
Associated Project: No
Associated Part Family: CY14V101xS
Software Version: None
Related Application Notes: AN43593

AN96589 discusses the Quad SPI (QSPI) and shows how to design with Cypress's 1-Mb QSPI nvSRAM. QSPI is an enhancement of the standard SPI protocol that provides four times the data throughput while maintaining the compact form factor of the standard serial SPI. System designs using QSPI devices occupy less board space and ultimately lower system costs.

Contents

1 Introduction

Cypress's 1-Mb QSPI nvSRAM is a high-performance nonvolatile SRAM product that offers truly random memory accesses (writes and reads). It is a monolithic integrated circuit with a quad SPI, which allows writing and reading the memory using a single (one I/O channel for one bit per clock cycle), dual (two I/O channels for two bits per clock cycle), or quad (four I/O channels for four bits per clock cycle) configuration via a function-rich command set..

The QSPI nvSRAM architecture incorporates Cypress's unique SRAM with silicon-oxide-nitride-oxide semiconductor (SONOS) nonvolatile elements. It blends the performance characteristics of a high-speed SRAM with a nonvolatile memory. The quad serial interface in QSPI nvSRAM conforms to the de facto industry-standard Quad Serial Peripheral Interface. The instruction set includes standard Quad SPI opcodes along with nvSRAM-specific functions and performance-oriented new features. The QSPI nvSRAM signals include SCK (serial clock), SI, and SO (for command/response and data input/output) and control signals CS#, HOLD#, and WP#. This hardware interface creates a low-pin-count device that reduces package size, PCB area, and overall system cost.

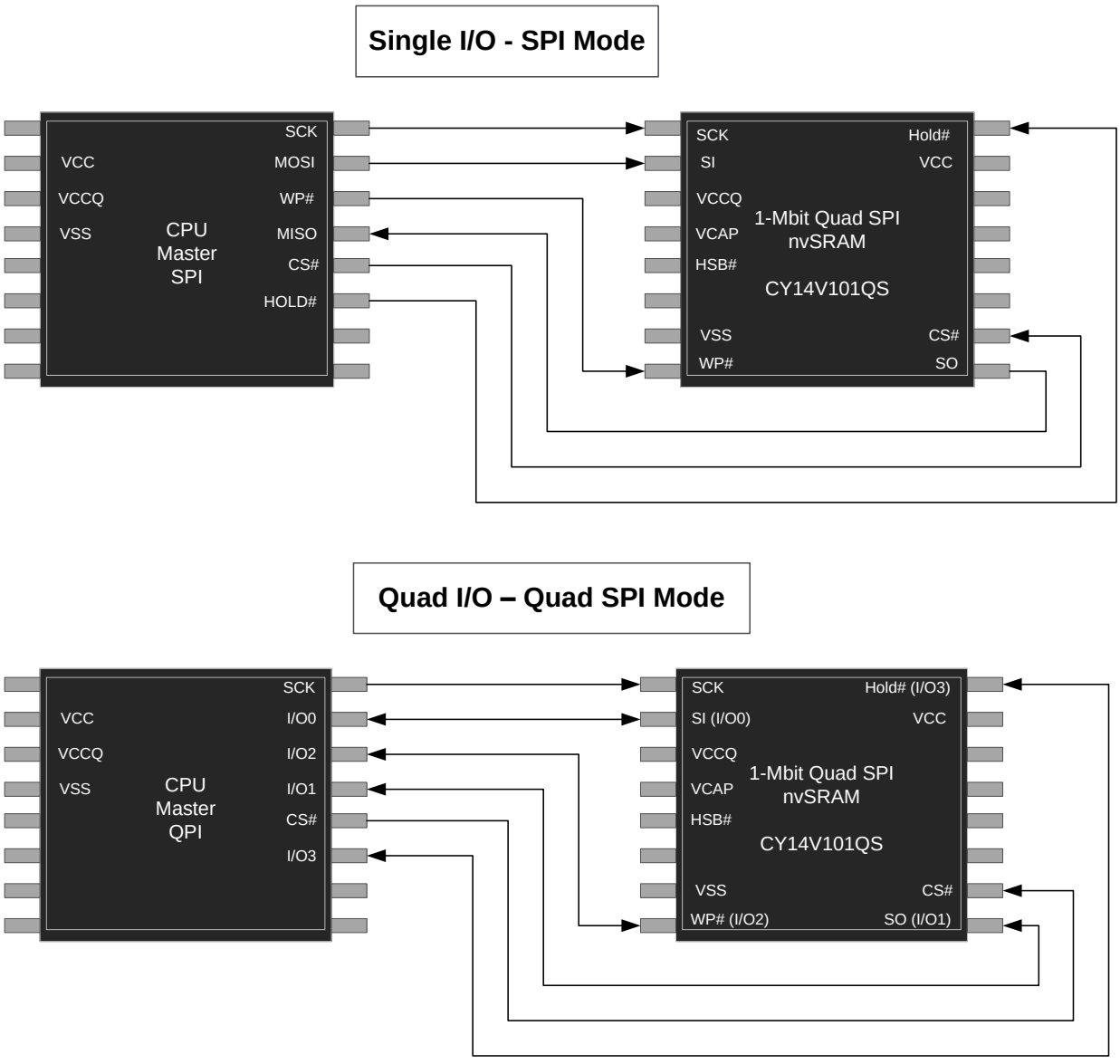
This application note discusses the QSPI and demonstrates how to use the 1-Mb QSPI nvSRAM in system designs.

2 QSPI Overview

QSPI is a communications protocol for interfacing a microcontroller (master) and one or more external memories (slaves). It is based on the popular Serial Peripheral Interface (SPI). SPI has four connections between the master and slave: Data In, Data Out, Clock (only uses the rising edge), and Chip Select (indicates which slave the SPI master is using). QSPI has six connections between the master and slave: four bidirectional serial data links, Clock (which uses either the rising edge or rising and falling edges), and Chip Select. The extra data links allow QSPI to have approximately four times more data throughput than SPI. With the high clock rate, the QSPI protocol can generate comparable data transfer rates to legacy parallel memory interfaces with 25 to 35 connections. Reducing the IC-to-IC connections by this factor of five requires instructions that mix address and data across the four serial channels.

Figure 1 shows the connectivity configuration with a host (Master SPI/QSPI) for 1-Mb QSPI nvSRAM in the SPI and QSPI configurations respectively.

Figure 1. SPI/QSPI Connectivity Overview



As can be seen in Figure 1, QSPI nvSRAM provides multiple I/O functionality by switching pin functions to support both a unidirectional and a bidirectional data bus. In SPI mode, the command, address, and data are serial using the SI and SO unidirectional channels for communication. In QSPI mode, the command, address, and data are also serial but use the SI (I/O0), SO (I/O1), WP# (I/O2), and Hold# (I/O3) bidirectional channels to serve x4 I/O communication. In QSPI mode, the hardware write protect and communication hold features are not present, as the WP# and HOLD# pins act as I/O2 and I/O3 respectively. QSPI nvSRAM also supports Dual SPI (two data channels) mode where SI (I/O0) and SO (I/O1) are used in bidirectional mode for command, address, and data communication. However, in Dual SPI mode, the hardware write protect and communication hold features are maintained.

Table 1 shows the w-x-y-z notation for each instruction supported by 1-Mb QSPI nvSRAM, where “w” specifies the number of channels for the command, “x” specifies the number of channels for the address, “y” specifies the number of channels for the mode/dummy, and “z” specifies the number of channels for data.

Table 1. w-x-y-z Instruction Set Notation

Instruction Description	Instruction Name	Opcode	SPI	DPI	QPI	SPI Extended
Control						
Write Disable	WRDI	04h	[1,-,-,-]	[2,-,-,-]	[4,-,-,-]	
Write Enable	WREN	06h	[1,-,-,-]	[2,-,-,-]	[4,-,-,-]	
Enable DPI	DPIEN	37h	[1,-,-,-]		[4,-,-,-]	
Enable QPI	QPIEN	38h	[1,-,-,-]	[2,-,-,-]		
Enable SPI	SPIEN	FFh		[2,-,-,-]	[4,-,-,-]	
Memory Read						
Read	READ	03h	[1,1,-,1]	[2,2,2,2]	[4,4,4,4]	
FastRead	FAST_READ	0Bh	[1,1,1,1]	[2,2,2,2]	[4,4,4,4]	
Dual Out (Fast) Read	DOR	3Bh				[1,1,1,2]
Quad Out (Fast) Read	QOR	6Bh				[1,1,1,4]
Dual IO (Fast) Read	DIOR	BBh				[1,2,2,2]
Quad IO (Fast) Read	QIOR	EBh				[1,4,4,4]
Memory Write						
Write	WRITE	02h	[1,1,-,1]	[2,2,-,2]	[4,4,-,4]	
Dual Input Write	DIW	A2h				[1,1,-,2]
Quad Input Write	QIW	32h				[1,1,-,4]
Dual IO Write	DIOW	A1h				[1,2,-,2]
Quad IO Write	QIOW	D2h				[1,4,-,4]
SR Commands						
Software Reset Enable	RSTEN	66h	[1,-,-,-]	[2,-,-,-]	[4,-,-,-]	
Software Reset	RESET	99h	[1,-,-,-]	[2,-,-,-]	[4,-,-,-]	
Read RTC	RDRTC	56h	[1,1,-,1]	[2,2,-,2]	[4,4,-,4]	
Write RTC	WRRTC	55h	[1,1,-,1]	[2,2,-,2]	[4,4,-,4]	
Fast Read RTC	FAST_RDRTC	57h	[1,1,1,1]	[2,2,2,2]	[4,4,4,4]	
Enter Hibernate mode	HIBEN	BAh	[1,-,-,-]	[2,-,-,-]	[4,-,-,-]	
Enter Sleep mode	SLEEP	B9h	[1,-,-,-]	[2,-,-,-]	[4,-,-,-]	
Exit Sleep mode	EXSLP	ABh	[1,-,-,-]	[2,-,-,-]	[4,-,-,-]	

Instruction Description	Instruction Name	Opcode	SPI	DPI	QPI	SPI Extended
Register Commands						
Read Status Register	RDSR	05h	[1,-,-,1]	[2,-,-,2]	[4,-,-,4]	
Write Status Register	WRSR	01h	[1,-,-,1]	[2,-,-,2]	[4,-,-,4]	
Read Configuration Register	RDCR	35h	[1,-,-,1]	[2,-,-,2]	[4,-,-,4]	
Write Configuration Register	WRCR	87h	[1,-,-,1]	[2,-,-,2]		
Read ID Register	RDID	9Fh	[1,-,-,1]	[2,-,-,2]	[4,-,-,4]	
Fast Read ID Register	FAST_RDID	9Eh	[1,-,1,1]	[2,-,2,2]	[4,-,4,4]	
Write Serial Number Register	WRSN	C2h	[1,-,-,1]	[2,-,-,2]	[4,-,-,4]	
Read Serial Number Register	RDSN	C3h	[1,-,-,1]	[2,-,-,2]	[4,-,-,4]	
Fast Read Serial Number Register	FAST_RDSN	C9h	[1,-,1,1]	[2,-,2,2]	[4,-,4,4]	
NV specific Commands						
STORE	STORE	8Ch	[1,-,-,-]	[2,-,-,-]	[4,-,-,-]	
RECALL	RECALL	8Dh	[1,-,-,-]	[2,-,-,-]	[4,-,-,-]	
AutoStore Enable	ASEN	8Eh	[1,-,-,-]	[2,-,-,-]	[4,-,-,-]	
AutoStore Disable	ASDI	8Fh	[1,-,-,-]	[2,-,-,-]	[4,-,-,-]	
Mode Bits						
Mode Bit (Set,Reset)		Axh, !Axh				

3 QSPI Instruction Protocol

All communication between the host and the 1-Mb QSPI nvSRAM is in the form of instructions. Each instruction begins with a command that selects the type of information transfer to be performed. Instructions may also have an address, command modifier (mode bits), latency (dummy cycles), and data transfer to or from the memory or registers. All instructions are executed serially between the host and 1-Mb QSPI nvSRAM.

The structure of the instructions is as follows:

- Each instruction begins with CS# going low and ends with CS# going high.
- SCK (serial clock) marks the transfer of each bit or group of bits.
- Each instruction begins with an 8-bit command. The command selects the type of device operation to be performed.
- The command may be standalone, or it may be followed by address bits to select a location within the memory address space or register map. The address is 24 bits for memory and 8 bits for registers.
- Some read instructions may require latency (dummy cycles) after the address bits. SCK continues to toggle, and data bits are driven at the end of the latency cycle.
- Some commands send a command modifier, called “mode bits,” following the address to indicate that the next instruction will be of the same type with an implied command. Thus, the next instruction will not provide a command byte, only a new address and mode bits. It is generally referred to as “execute in place (XIP).”
- Write or read data follows the address or mode bits.
- CS# must go high after the instruction. For instructions that do not return data, the CS# signal must go high after the eighth bit. For write instructions, CS# must go high after the last data byte that is transferred. Otherwise, the command is rejected.

- SCK continues toggling while CS# is low during any read/write instruction that executes the burst data mode. In this mode, the address is incremented automatically inside the device while data is continuously shifted in or out of the device.
- All the commands, the address, and mode bits are shifted into the device with the most-significant bits first. The data bits are shifted in or out of the device with the most significant bits first as well.
- All attempts to access the device during STORE/RECALL operations are ignored.
- Depending on the instruction, the execution time varies. The Read Status Register instruction can be used to determine if the device is busy and whether the instruction has completed execution.

3.1 Command Sequence Examples

The following sections provide sequences for different instructions. The timing details for each command are specified in the Cypress 1-Mb QSPI nvSRAM datasheets.

3.1.1 READ Instruction – Access the Memory Array

Figure 2 shows the memory read instruction sequence in SPI mode. Here, if CS# is not brought high after the first read data byte and SCK is continuously toggled, burst data mode is executed. To end the instruction, CS# must be brought high.

Figure 2. READ Functional Flow Chart – SPI Mode

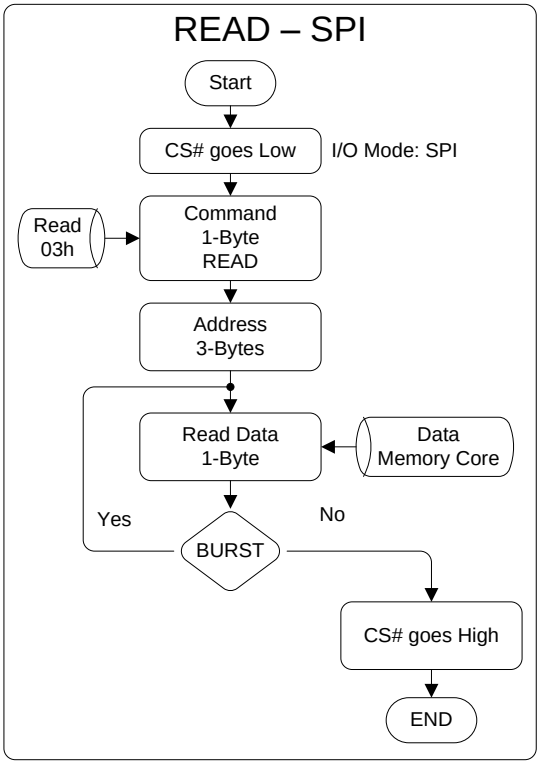


Figure 3 shows the memory read instruction sequence in Dual SPI mode. It is assumed that the device starts in SPI mode, enables Dual SPI mode, and is brought back to SPI mode after the instruction sequence. Here, a single cycle of latency is required after the address bits. Again, if CS# is kept low after the first read data byte and SCK is continuously toggled, the burst data mode is executed. To end the instruction, CS# must be brought high.

Figure 3. READ Functional Flow Chart – Dual SPI Mode

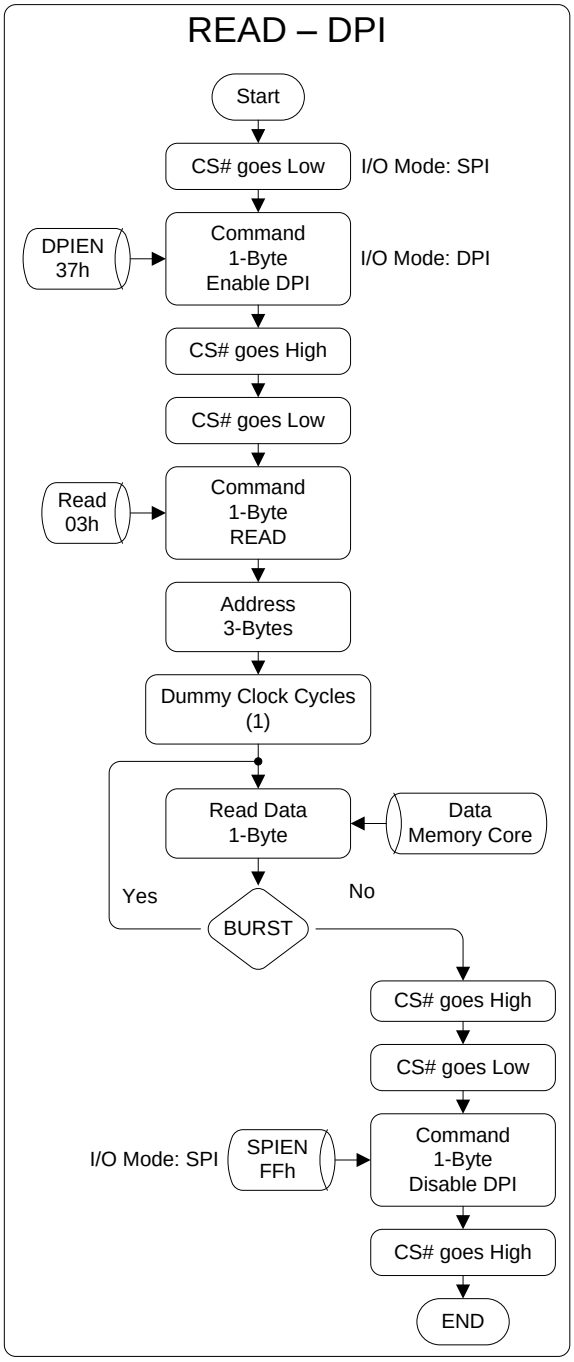
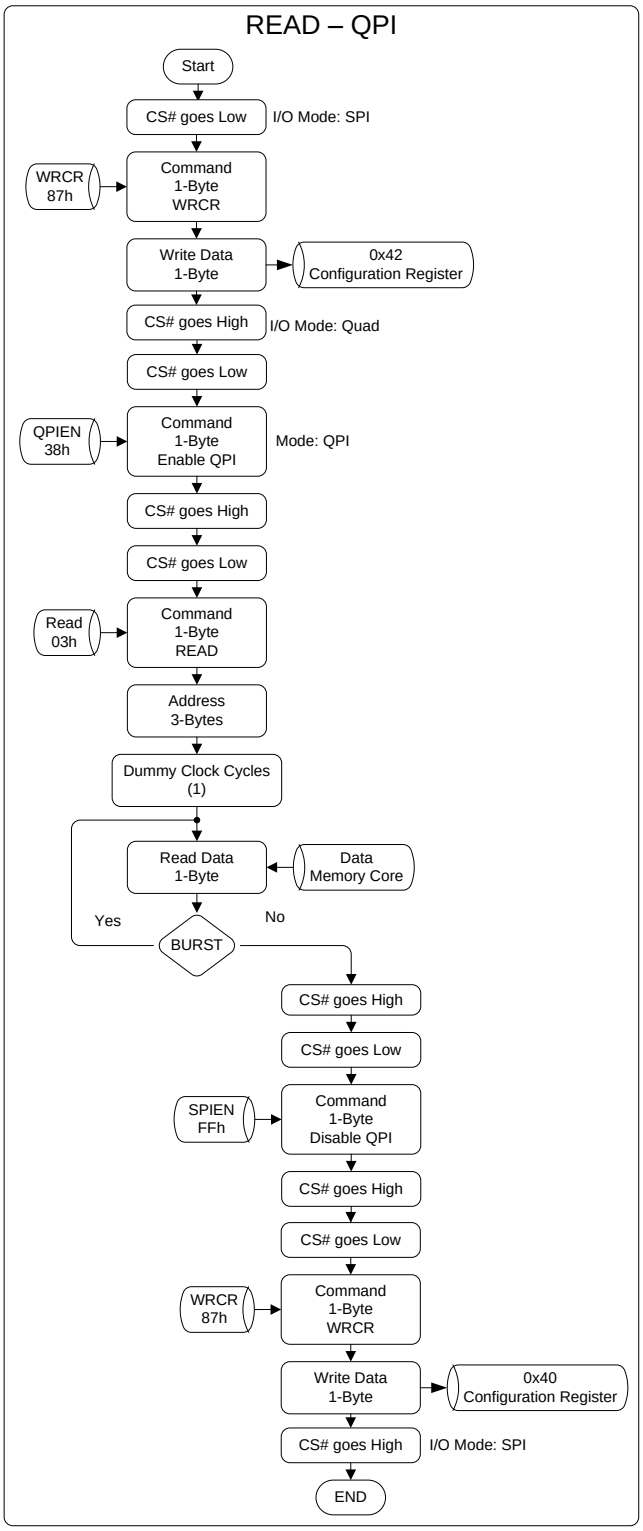


Figure 4 shows the memory read instruction sequence in QSPI mode. It is assumed that the device starts in SPI mode, enables QSPI mode, and is brought back to SPI mode after the instruction sequence. Here, a single cycle of latency is again required after the address bits. If CS# is not brought high after the first read data byte and SCK is continuously toggled, the burst data mode is executed. To end the instruction, CS# must be brought high.

Figure 4. READ Functional Flow Chart – QSPI Mode



3.1.2 FAST_READ Instructions – Access the Memory Array

Figure 5 shows the memory fast read instruction sequence in SPI mode. The fast read instruction can be executed up to 108 MHz. Here, a mode byte option is shown after the address bits to determine whether XIP is being carried out. Also, if CS# is not brought high after the first read data byte and SCK is continuously toggled, the burst data mode is executed. To end the instruction, CS# must be brought high.

Figure 5. FAST_READ Functional Flow Chart – SPI Mode

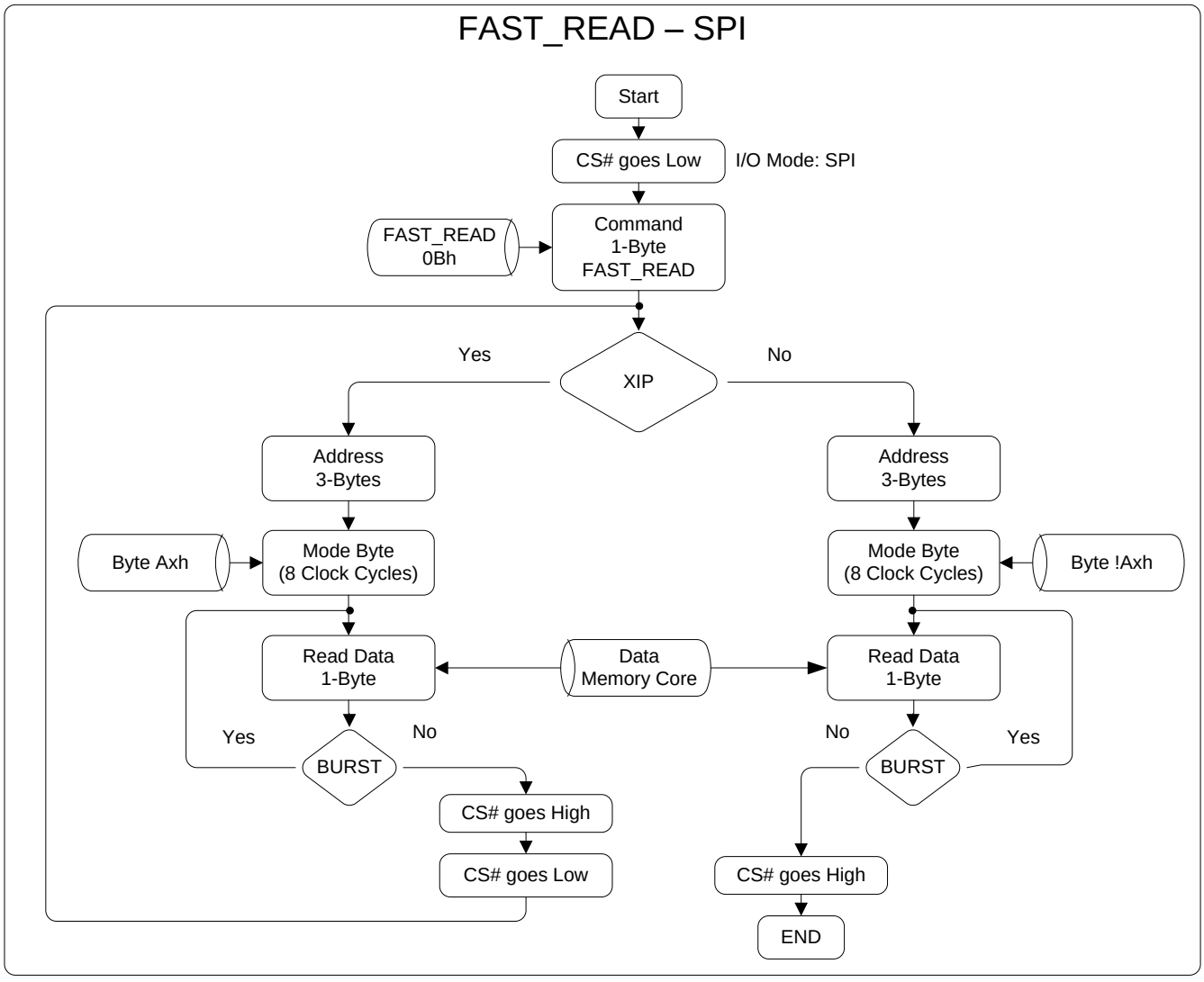


Figure 6 shows the memory fast read instruction sequence in Dual SPI mode. It is assumed that the device starts in SPI mode, enables Dual SPI mode, and is brought back to SPI mode after the instruction sequence. Here, again a mode byte option is shown after the address bits to determine whether XIP is being carried out. Also, if CS# is not brought high after the first read data byte and SCK is continuously toggled, the burst data mode is executed. To end the instruction, CS# must be brought high.

Figure 6. FAST_READ Functional Flow Chart – Dual SPI Mode

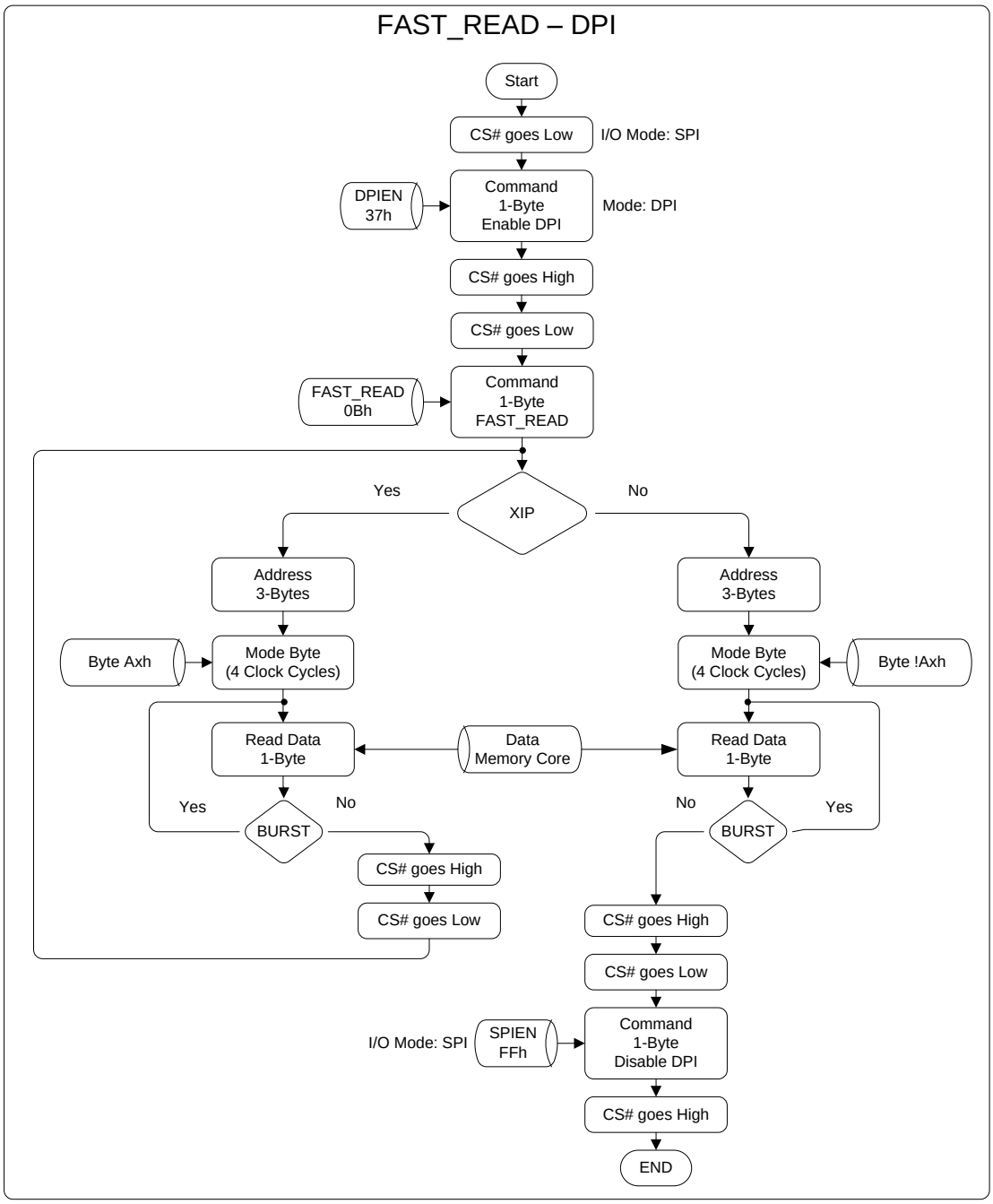
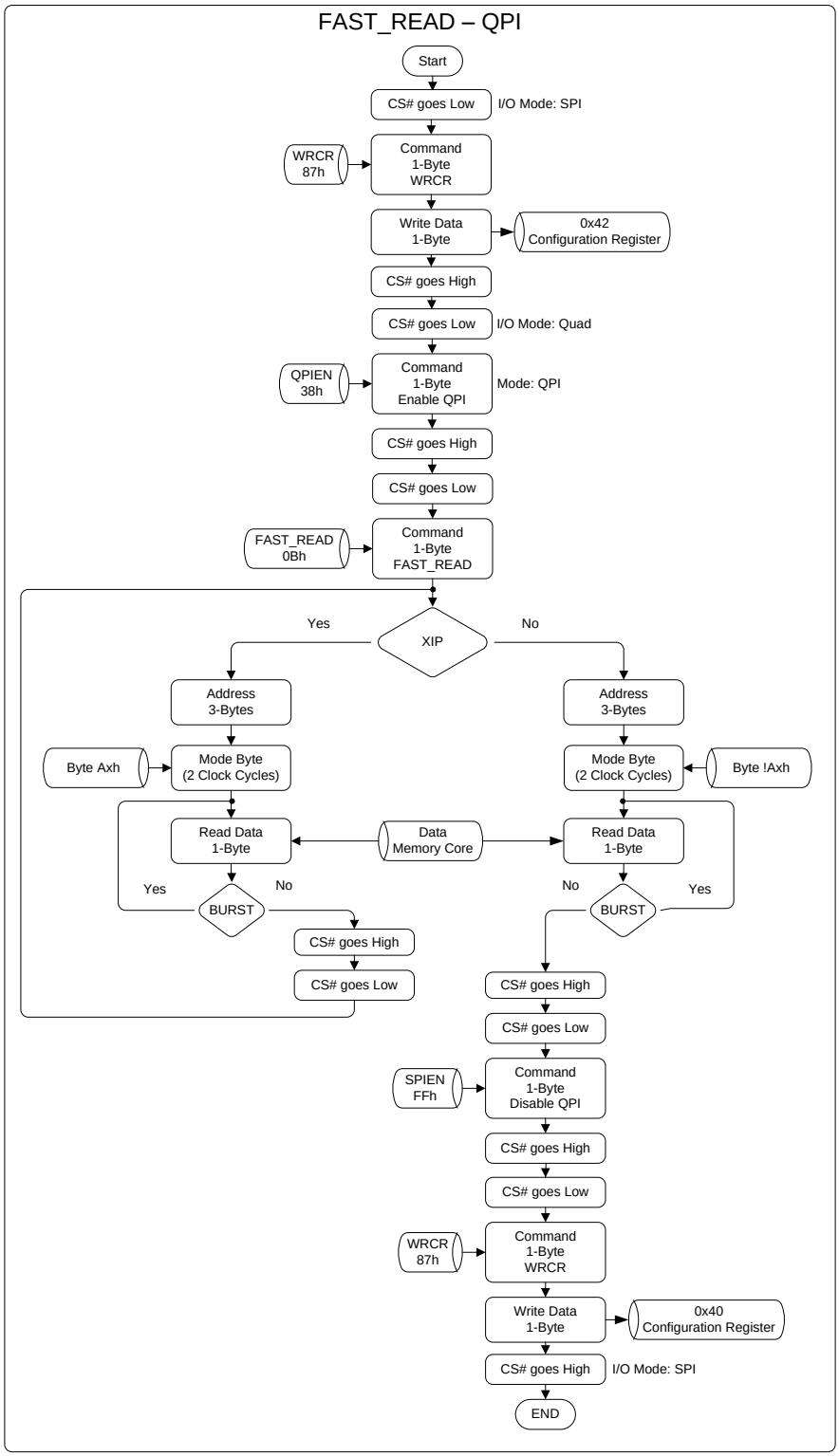


Figure 7 shows the memory fast read instruction sequence in QSPI mode. It is assumed that the device starts in SPI mode, enables QSPI mode, and is brought back to SPI mode after the instruction sequence. Here, again a mode byte option is shown after the address bits to determine whether XIP is being carried out. Also, if CS# is not brought high after the first read data byte and SCK is continuously toggled, the burst data mode is executed. To end the instruction, CS# must be brought high.

Figure 7. FAST_READ Functional Flow Chart – QSPI Mode



3.1.3 WRITE Instruction – Access the Memory Array

Figure 8 shows the memory write instruction sequence in SPI mode. First, Write Enable Latch (WEL) must be set before any WRITE operation to the memory can be performed. Afterward, if CS# is not brought high following the first write data byte and SCK is continuously toggled, the burst data mode is executed. Finally, WEL should be disabled since a write command to memory does not reset WEL. To end the instruction, CS# must be brought high.

Figure 8. WRITE Functional Flow Chart – SPI Mode

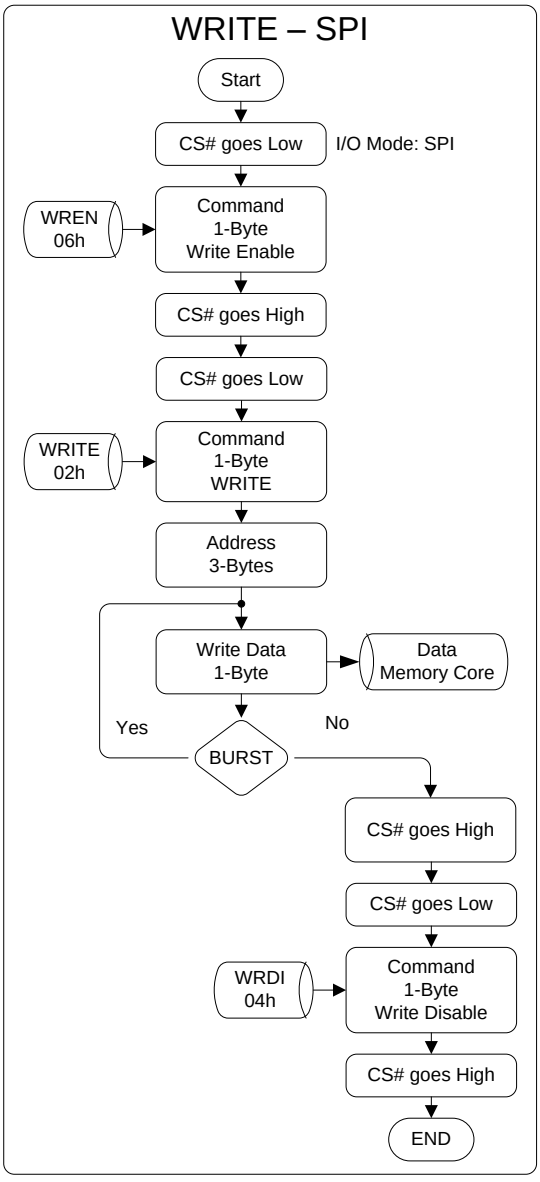


Figure 9 shows the memory write instruction sequence in Dual SPI mode. It is assumed that the device starts in SPI mode, enables Dual SPI mode, and is brought back to SPI mode after the instruction sequence. First, Write Enable Latch (WEL) must be set before any WRITE operation to the memory can be performed. Afterward, if CS# is kept low after the first write data byte and SCK is continuously toggled, the burst data mode is executed. Finally, WEL should be disabled since a write command to memory does not reset WEL. To end the instruction, CS# must be brought high.

Figure 9. WRITE Functional Flow Chart – Dual SPI Mode

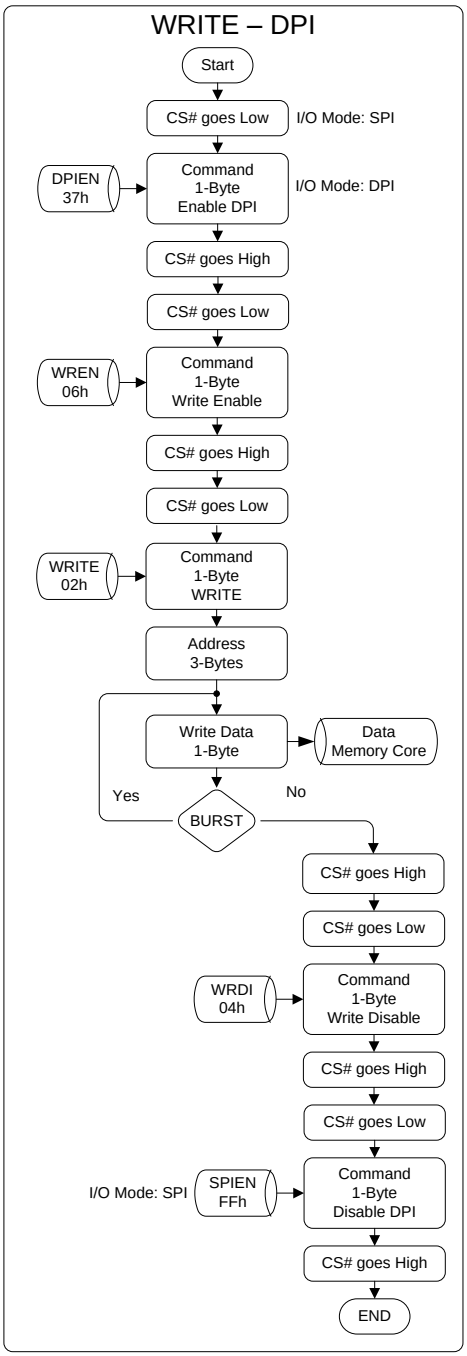
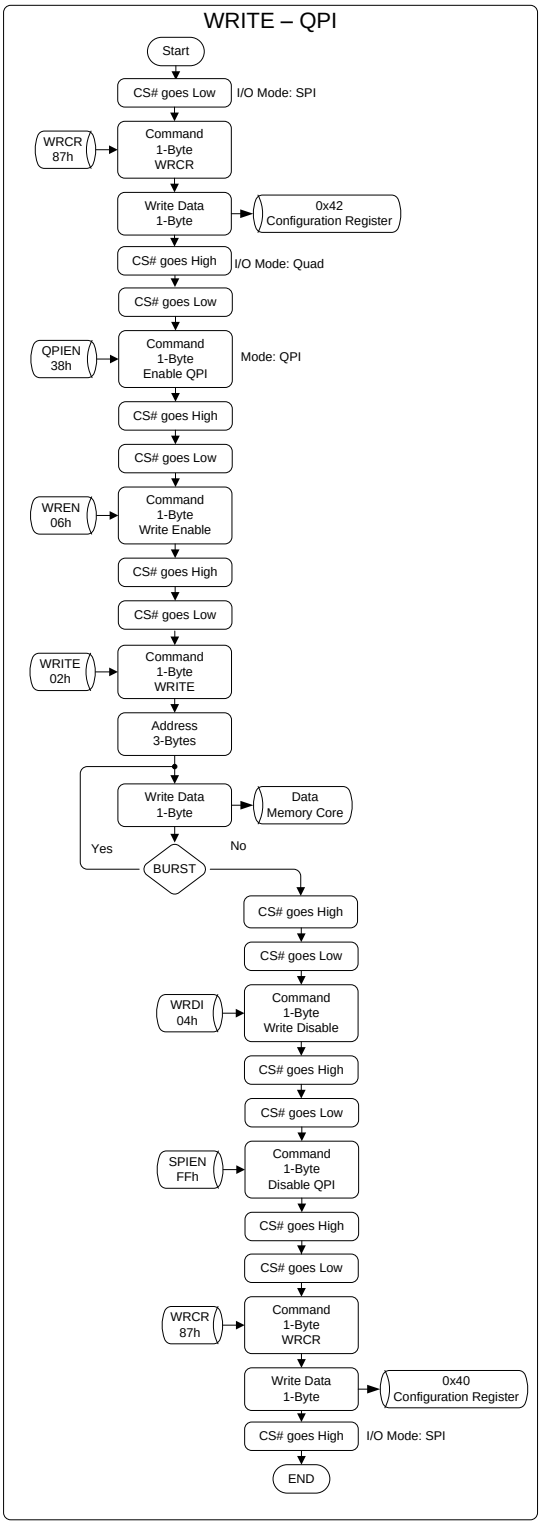


Figure 10 shows the memory write instruction sequence in QSPI mode. It is assumed that the device starts in SPI mode, enables QSPI mode, and is brought back to SPI mode after the instruction sequence. First, Write Enable Latch (WEL) must be set before any WRITE operation to the memory can be performed. Afterward, if CS# is not brought high after the first write data byte and SCK is continuously toggled, the burst data mode is executed. Finally, WEL will need to be disabled since a write command to memory does not reset WEL to avoid any inadvertent writes. To end the instruction, CS# must be brought high.

Figure 10. WRITE Functional Flow Chart – QSPI Mode



3.1.4 HIBEN Instruction – Power Save Hibernate Mode

Figure 11 shows the hibernate instruction sequence in SPI mode. Hibernate mode is the lowest power mode with a device leakage current of I_{ZZ} (10 µA). The hibernate instruction initiates a STORE operation (if a write instruction has been executed before the last STORE operation) in which data is transferred from the SRAM elements to the nonvolatile SONOS memory elements. The HIBEN operation takes t_{HIBEN} (8 ms), and the status register WIP (Work In Progress) bit can be polled to determine the completion status. To end the instruction, CS# must be brought high.

Figure 11. HIBEN Functional Flow Chart – SPI Mode

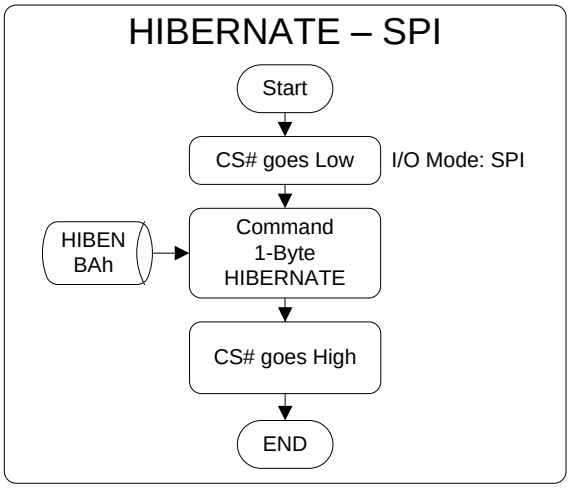
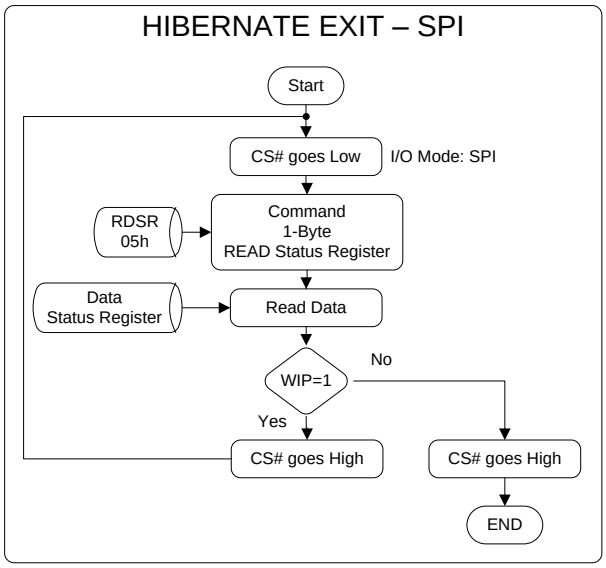


Figure 12 shows the hibernate exit instruction sequence in SPI mode. Exit from hibernate begins by bringing CS# low, and it requires a RECALL operation where data is transferred from the nonvolatile SONOS memory elements to SRAM elements. The HIBERNATE EXIT operation takes t_{WAKE} (20 ms), and the status register WIP bit can be polled to determine the completion status. To end the instruction, CS# must be brought high.

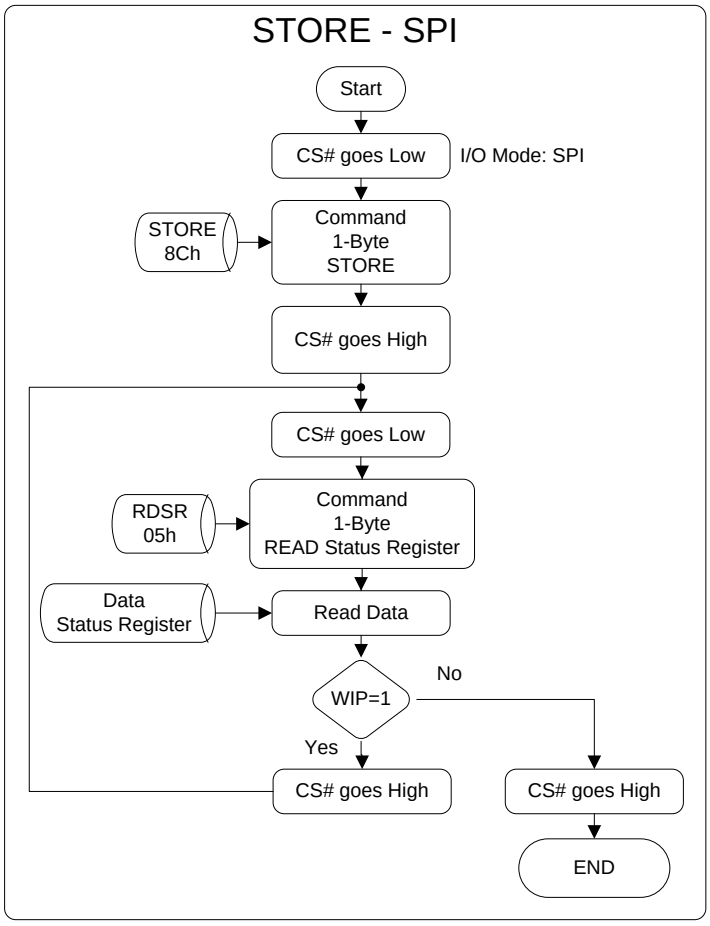
Figure 12. HIBERNATE EXIT Functional Flow Chart – SPI Mode



3.1.5 STORE Instruction – Nonvolatile Store Operation

Figure 13 shows the store instruction sequence in SPI mode. The STORE operation transfers data from SRAM elements to SONOS nonvolatile elements in QSPI nvSRAM. The completion status of STORE can be checked by polling the WIP bit in the status register. QSPI nvSRAM also provides a FAULT register, where checking the STORE bit once WIP goes low shows successful completion. To end the instruction, CS# must be brought high.

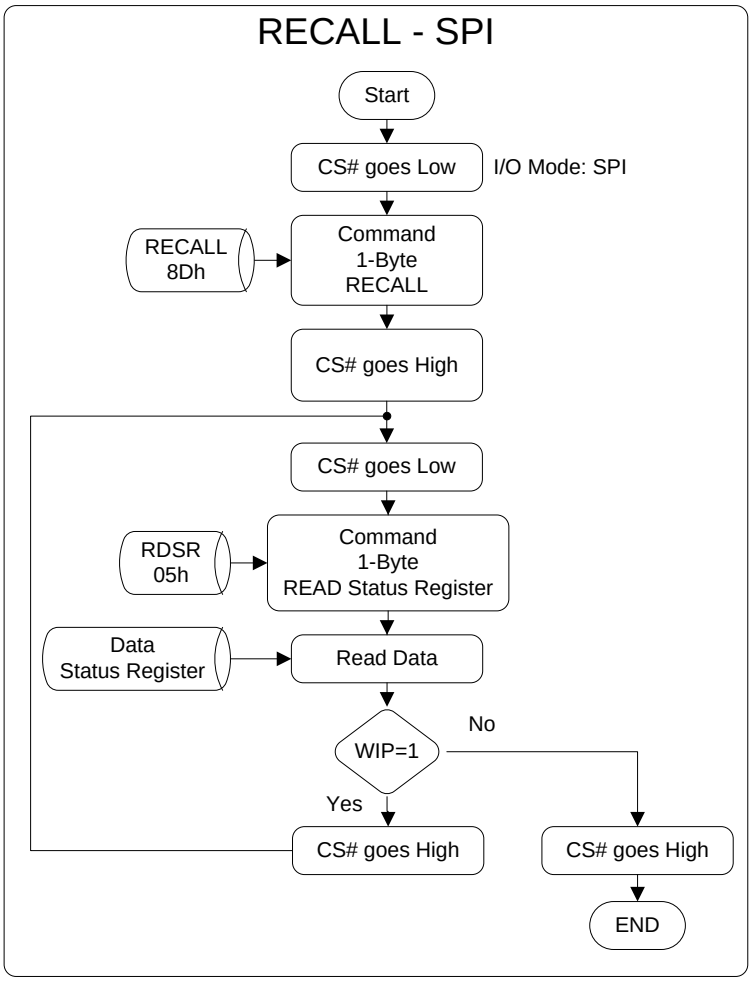
Figure 13. STORE Functional Flow Chart – SPI Mode



3.1.6 RECALL Instruction – Nonvolatile Recall Operation

Figure 14 shows the recall instruction sequence in SPI mode. The RECALL operation transfers data from SONOS nonvolatile elements to SRAM elements in Quad SPI nvSRAM. The completion status of RECALL can be checked by polling the WIP bit in the status register. QSPI nvSRAM also provides a FAULT register, where checking the RECALL bit once WIP goes low shows successful completion. To end the instruction, CS# must be brought high.

Figure 14. RECALL Functional Flow Chart – SPI Mode



4 PCB Layout Guides

This section provides general layout recommendations.

4.1 Power Supply Decoupling

The Cypress 1-Mb QSPI nvSRAM has two power supply input pins (VCC and VCCQ) and one ground pin (VSS). The use of one 0.1-µF ceramic capacitor, normally in a 0603 or 0402 package, is recommended for decoupling each power supply input pin. This decoupling capacitor should be placed as close as possible to the power supply input pins. The routing of the decoupling capacitors should be optimized to achieve low inductance. Power supply trace lengths from the package pads to the vias should be as short as possible with a trace width of approximately 0.6 mm. It is recommended that you avoid sharing the same via with two or more decoupling capacitors.

4.2 Clock Routing

For reliable high-speed synchronous data transfers, it is essential that the clock signal have very good signal integrity. Following are recommendations for routing the clock signal:

- Run the clock signal at least 3x of the trace width away from all other signal traces. This will help keep the clock signal clean from noise.
- Use as few vias as possible for the entire path of the clock signal. Each via creates impedance changes and may cause signal reflections.
- Run the clock trace as straight as possible and avoid using serpentine routing.
- Keep a continuous ground in the next layer as a reference plane.
- Route the clock trace with controlled impedance, typically a 50-Ω trace impedance with ± 5 percent tolerance.

4.3 Data Signal Routing

The Cypress 1-Mb QSPI nvSRAM supports 1-bit, 2-bit, and 4-bit data bus configurations. In 2-bit and 4-bit multiple I/O configurations, it is important that the I/O traces are routed such that they have identical lengths, within approximately 1 mm, to ensure equivalent propagation delays. To promote reliable data transfers for all configurations, make sure that the propagation delays for the clock trace and all data traces are identical. The data signals should be routed with traces of controlled impedance, typically 50 Ω, to reduce signal reflection. Data traces should have no 90° angle corners. The preferred method for implementing a 90° angle change is to cut the corner to smooth the trace. To maximize signal integrity, avoid using multiple signal layers for data signal routing and ensure all signal traces have a continuous reference plane.

5 Summary

AN96589 provided an overview of the QSPI. It introduced the instruction set and instruction protocol and then provided a series of communication examples between a host (master) and Cypress's 1-Mb QSPI nvSRAM (slave) device. It also included recommendations for the PCB layout to achieve optimum performance.

Document History

Document Title: AN96589 - Designing with Cypress 1-Mb Quad SPI nvSRAM

Document Number: 001-96589

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	4703103	SZZX	03/27/2015	New Spec.
*A	4829487	SZZX	07/09/2015	Incorporated Instruction Opcode changes Updated template

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Automotive	cypress.com/go/automotive
Clocks & Buffers	cypress.com/go/clocks
Interface	cypress.com/go/interface
Lighting & Power Control	cypress.com/go/powerpsoc
Memory	cypress.com/go/memory
PSoC	cypress.com/go/psoc
Touch Sensing	cypress.com/go/touch
USB Controllers	cypress.com/go/usb
Wireless/RF	cypress.com/go/wireless

PSoC® Solutions

psoc.cypress.com/solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/go/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com