QSPI Driver API Reference

Technical Note

FPGA-TN-02339-1.4

June 2025

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUNDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Inclusive Language

This document was created consistent with Lattice Semiconductor’s inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice’s inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Figures

Tables

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
API	Application Programming Interface
QSPI	Quad Serial Peripheral Interface
FIFO	First In, First Out
SPI	Serial Peripheral Interface
AXI	Advanced extensible Interface
CRC	Cyclic Redundancy Check
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
MAC	Media Access Controller
SDK	Software Development Kit
TSEMAC	Tri-Speed Ethernet Media Access Controller
WEL	Write Enable Latch

1. Introduction

The Quad Serial Peripheral Interface (QSPI) is a four-tri-state data line serial interface that is commonly used to program, erase, and read SPI flash memories. QSPI enhances the throughput of a standard SPI by four times as four bits are transferred every clock cycle.

Refer to the QSPI Flash Controller IP User Guide (FPGA-IPUG-02248) for more details about the IP core.

1.1. Purpose

This document is intended to act as a reference guide for developers by providing details of the C language driver APIs and function call flows.

1.2. Audience

The intended audience for this document includes embedded system designers and embedded software developers using Lattice CertusPro™-NX and Lattice Avant™-E devices. The technical guide assumes readers have expertise in embedded systems and FPGA technologies.

1.3. Driver Version

QSPI_FLASH_CONTROLLER_DRV_VER “25.01.00”

1.4. Driver and IP Compatibility

Driver version	IP version
25.01.00	1.5.0

Driver tested on hardware device	Tools version
CertusPro-NX Versa Evaluation Board Avant-E Evaluation Board (Revision D)	Lattice Propel™ Builder 2024.2 Lattice Propel SDK 2024.2 Lattice Radiant™ Software 2024.2 Radiant Programmer 2024.2

Refer to the QSPI Flash Controller IP Release Notes (FPGA-RN-02016) for more information on the driver and IP versions.

2. API Description

2.1. print_config_reg()

This API is used to print configuration register values according to the input parameters.

```
void print_config_reg(config_output_type_t dbg_val)
```

In/Out	Parameter	Description	Returns
In	dbg_val	Parameter to indicate debug output level.	None

2.2. qspi_trans_start()

This API is used to generate SPI transaction and start QSPI flash operation.

```
unsigned char qspi_trans_start()
```

In/Out	Parameter	Description	Returns
None	None	None	0: success 1: failure

2.3. qspi_flash_cntl_init()

This API is used to initialize the base address of the QSPI flash controller.

```
unsigned int qspi_flash_cntl_init(struct qspi_flash_cntl_instance_t *this_qspi_flash_cntl,
)
```

In/Out	Parameter	Description	Returns
In	this_qspi_flash_cntl	Handle of the qspi_flash_cntl_instance_t structure.	0: success 1: failure

2.4. qspi_flash_cntl_read()

This API is used to read the data from the registers specified by *index* and stores the result in the *reg_data* pointer.

```
unsigned int qspi_flash_cntl_read(struct qspi_flash_cntl_instance_t *this_qspi_flash_cntl,
qspi_reg_type_t index , unsigned int *reg_data)
```

In/Out	Parameter	Description	Returns
In	this_qspi_flash_cntl	Handle of the qspi_flash_cntl_instance_t structure.	0: success 1: failure
In	index	Index for the register to data read the data from.	
Out	reg_data	Variable address where the read register value is stored.	

2.5. qspi_flash_cntl_write_fifo_dis()

This API is used to write the content from buffer into SPI flash with specified start address and length.

```
unsigned char qspi_flash_cntl_write_fifo_dis(unsigned int flash_addr, unsigned int
flash_addr_width, unsigned int spi_mode, unsigned int *buffer, unsigned int buffer_len)
```

In/Out	Parameter	Description	Returns
In	flash_addr	Flash start address to be written.	0: success 1: failure
In	flash_addr_width	Flash address width.	
In	spi_mode	SPI mode to use in write command.	
In	*buffer	Pointer of the buffer with data to be written.	
In	buffer_len	Buffer length of the data to be written.	

2.6. qspi_flash_cntl_read_id()

This API is used to read the ID from the SPI flash controller.

```
unsigned char qspi_flash_cntl_read_id(unsigned int *buffer)
```

In/Out	Parameter	Description	Returns
In	*buffer	Buffer to store the ID value.	0: success 1: failure

2.7. qspi_flash_cntl_erase()

This API is used to perform data erase from the flash memory with the given flash address.

```
unsigned char qspi_flash_cntl_erase(unsigned int flash_addr, unsigned int flash_addr_width,
unsigned int spi_mode, unsigned int erase_type)
```

In/Out	Parameter	Description	Returns
In	flash_addr	Flash address to perform erase.	0: success 1: failure
In	flash_addr_width	Flash address width to perform erase.	
In	spi_mode	SPI mode to perform erase.	
In	erase_type	Erase type to use.	

2.8. qspi_flash_cntl_write_erase_fifo_en()

This API is used to write data to or erase data from the flash memory to the given flash address using the TX FIFO.

```
unsigned int qspi_flash_cntl_write_erase_fifo_en(qspi_params_t *this_qspi_fifo, unsigned
int *buffer)
```

In/Out	Parameter	Description	Returns
In	this_qspi_fifo	Handle of the qspi_params_t structure.	0: success 1: failure
In	buffer	Buffer that holds the data to be sent to the TX FIFO.	

2.9. qspi_flash_cntl_read_fifo_en()

This API is used to read the data from the flash memory from the given flash address using RX FIFO.

```
void  qspi_flash_cntl_read_fifo_en(qspi_params_t *this_qspi_fifo, unsigned int *buffer)
```

In/Out	Parameter	Description	Returns
In	this_qspi_fifo	Handle of the qspi_params_t structure.	None
Out	buffer	Buffer to store the data after reading from RX FIFO.	

2.10. qspi_flash_cntl_read_fifo_dis()

This API is used to read the data from the flash memory from the given flash address using the Data Packet register.

```
unsigned char qspi_flash_cntl_read_fifo_dis(unsigned int flash_addr, unsigned int
flash_addr_width, unsigned int spi_mode,unsigned int *buffer, unsigned int
buffer_len,unsigned int manufacturer_id)
```

In/Out	Parameter	Description	Returns
In	flash_addr	Starting flash address to read from	0: success 1: failure
In	flash_addr_width	Flash address width	
In	spi_mode	SPI mode to use (dual, quad)	
Out	buffer	Buffer to store the data read from the Data Packet register.	
In	buffer_len	Buffer length in Header 0	
In	manufacturer_id	Manufacturer ID	

2.11. qspi_flash_cntl_set_pkt_hdr()

This API is used to configure the packet header register values when FIFO is disabled.

```
void qspi_flash_cntl_set_pkt_hdr(qspi_params_t *this_pkt_hdr)
```

In/Out	Parameter	Description	Returns
In	this_pkt_hdr	Handle of the qspi_params_t structure.	None

2.12. qspi_flash_cntl_set_tx_fifo()

This API is used to set the TX FIFO register values when FIFO is enabled.

```
void qspi_flash_cntl_set_tx_fifo(qspi_params_t *this_tx_fifo)
```

In/Out	Parameter	Description	Returns
In	this_tx_fifo	Handle of the qspi_params_t structure.	None

2.13. quad_mode()

This API is used to set up the necessary registers to enable and disable Quad mode for the QSPI flash controller.

```
unsigned char quad_mode(unsigned short en_mode)
```

In/Out	Parameter	Description	Returns
In	en_mode	Parameter to enter or exit quad mode.	0: success 1: failure

2.14. latch_set_clear()

This API is used to set the write enable latch (WEL) bit before other commands to change data.

```
unsigned char latch_set_clear(unsigned short spi_mode, unsigned short en_mode)
```

In/Out	Parameter	Description	Returns
In	spi_mode	SPI mode to use.	0: success 1: failure
In	en_mode	Parameter to enable or disable WEL mode.	

2.15. four_byte_mode()

This API is used to enter or exit four-byte mode.

```
unsigned char four_byte_mode(unsigned short spi_mode, unsigned short en_mode)
```

In/Out	Parameter	Description	Returns
In	spi_mode	SPI mode to use.	0: success 1: failure
In	en_mode	Parameter to enter or exit four-byte address mode.	

2.16. qspi_flash_cntl_read_write_config_reg()

This API is used to read or write the SPI flash configuration registers.

```
void qspi_flash_cntl_read_write_config_reg(unsigned short mode, unsigned int config)
```

In/Out	Parameter	Description	Returns
In	mode	Read = 0, Write = 1	Value returned from SPI flash
In	config	Config value to write in DATA0.	

3. Function Call Flow Diagrams

3.1. print_config_reg()

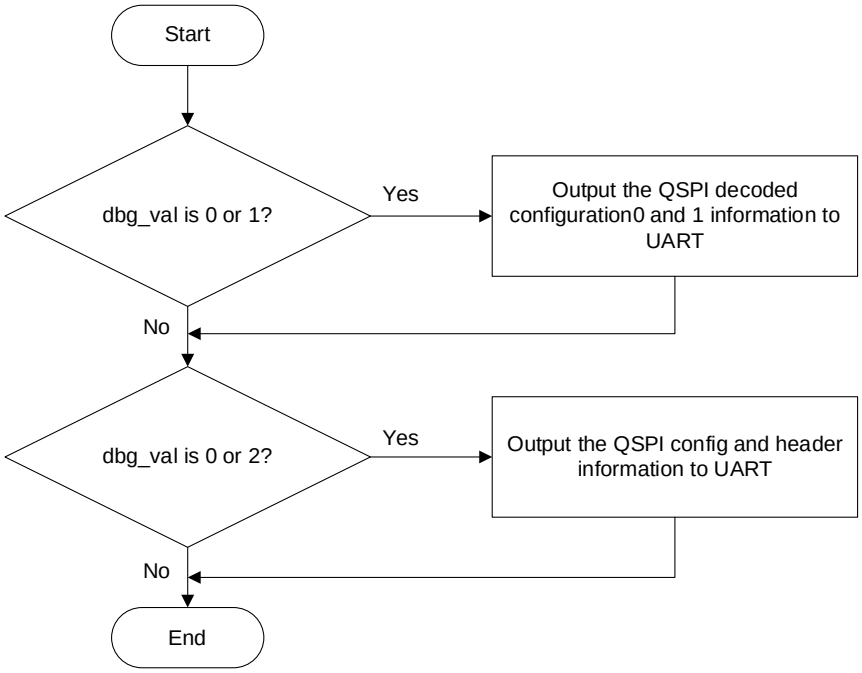

Figure 3.1. void print_config_reg()

3.2. qspi_trans_start()

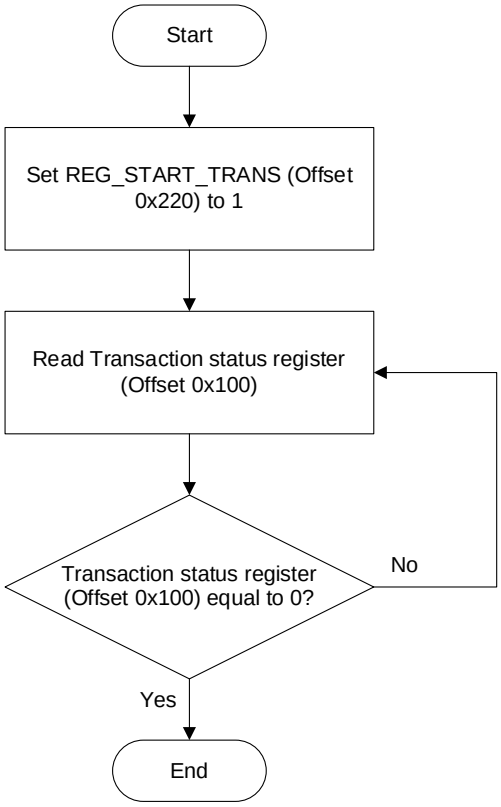

Figure 3.2. unsigned char qspi_trans_start()

3.3. qspi_flash_cntl_init()

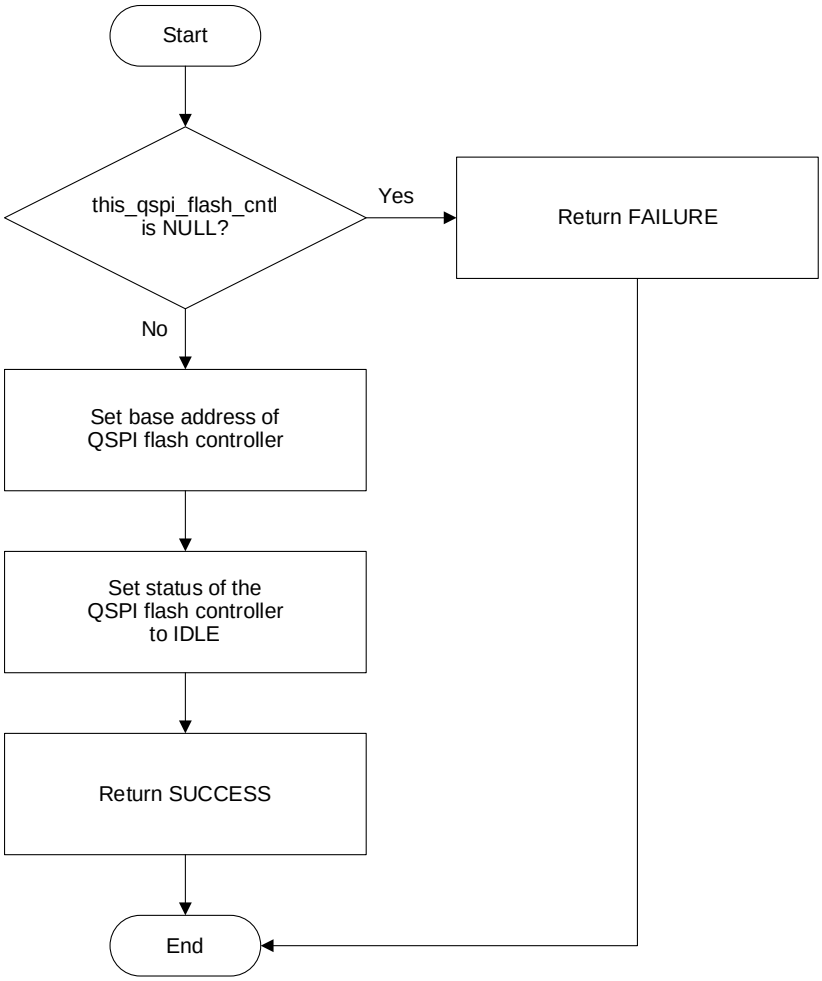

Figure 3.3. unsigned int qspi_flash_cntl_init()

3.4. qspi_flash_cntl_read()

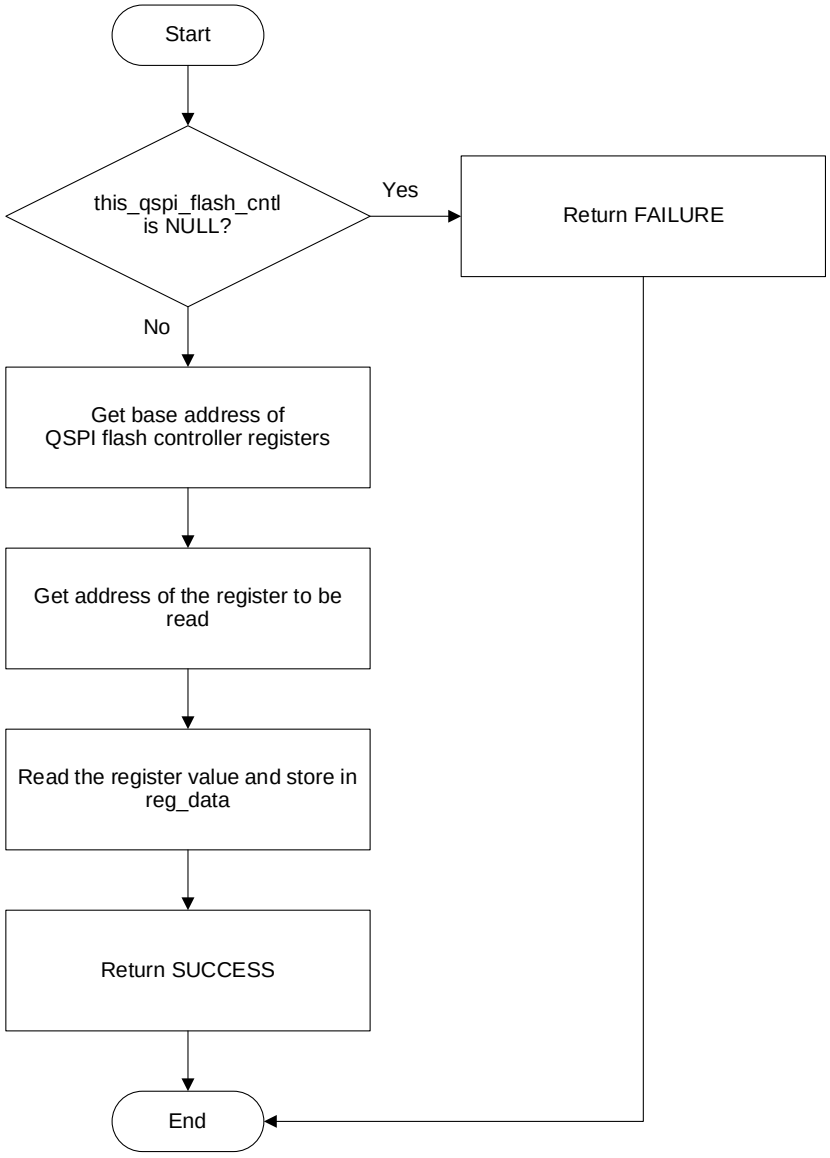

Figure 3.4. unsigned int qspi_flash_cntl_read()

3.5. qspi_flash_cntl_write_fifo_dis()

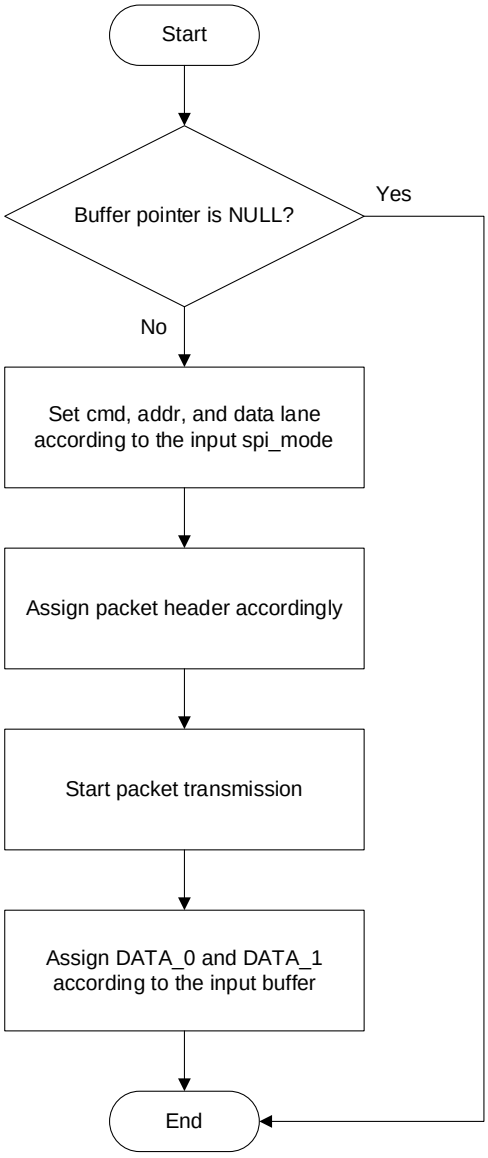

Figure 3.5. unsigned char qspi_flash_cntl_write_fifo_dis()

3.6. qspi_flash_cnt_read_id()

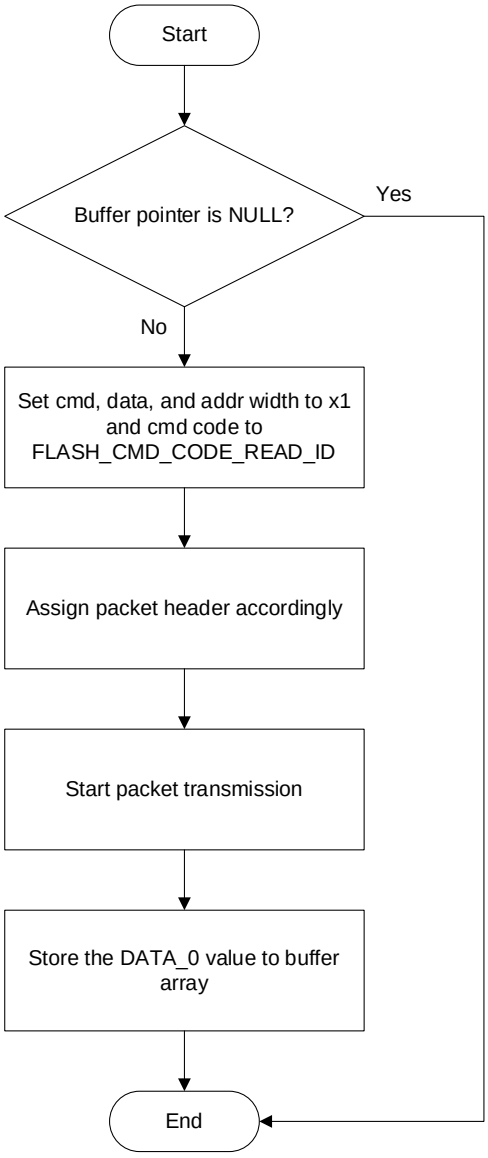

Figure 3.6. unsigned char qspi_flash_cnt_read_id()

3.7. qspi_flash_cntl_erase()

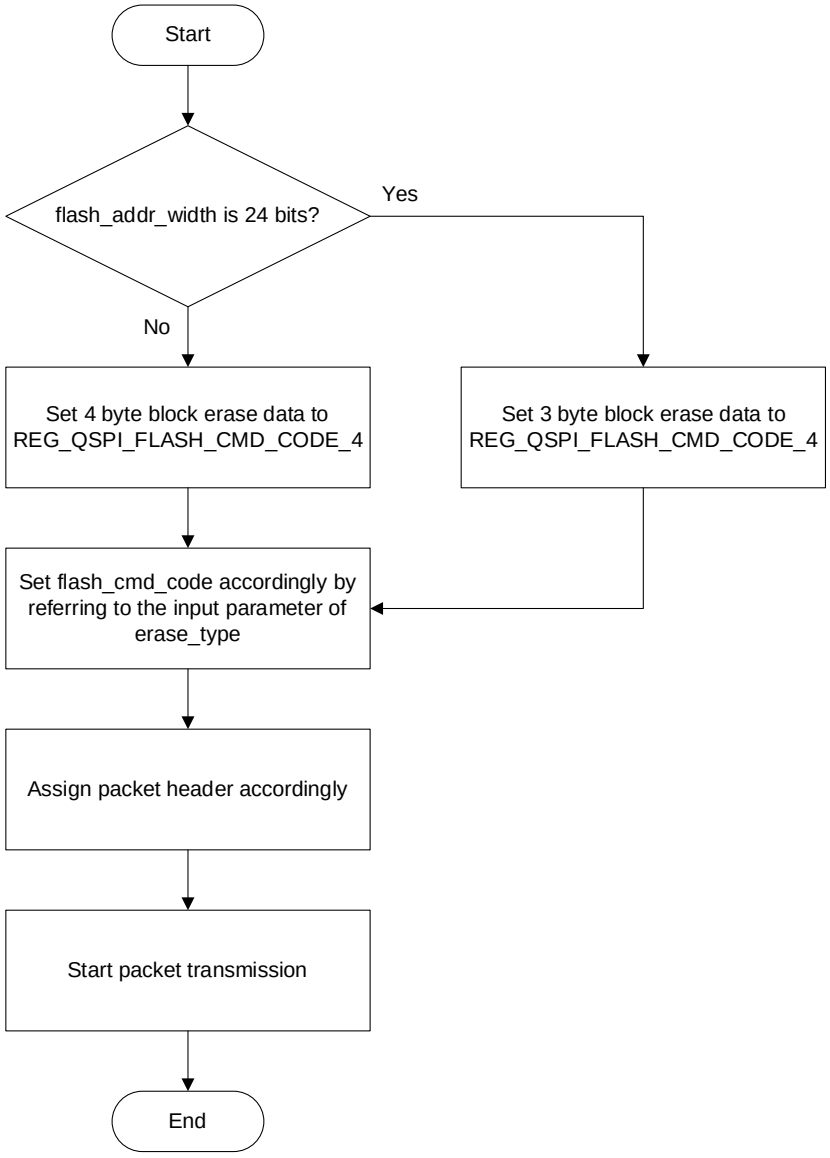

Figure 3.7. unsigned char qspi_flash_cntl_erase()

3.8. qspi_flash_cntl_write_erase_fifo_en()

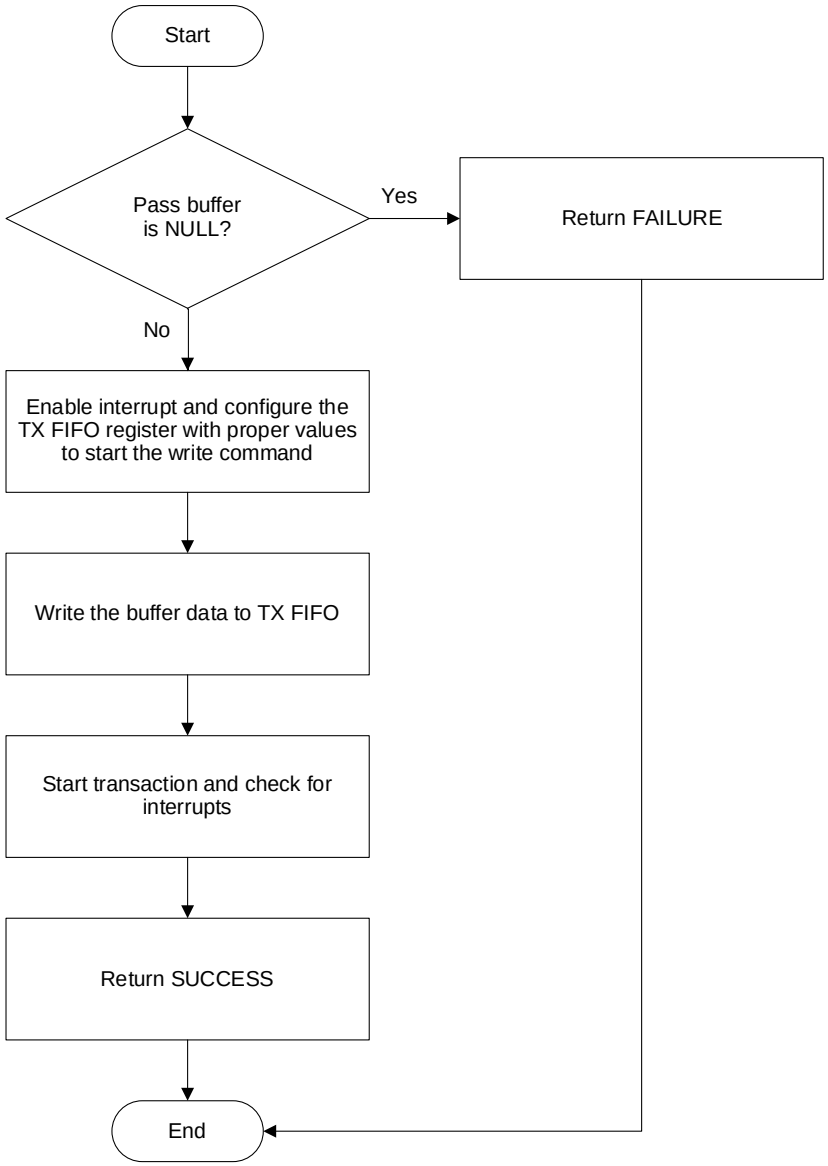

Figure 3.8. unsigned int qspi_flash_cntl_write_erase_fifo_en()

3.9. qspi_flash_cntl_read_fifo_en()

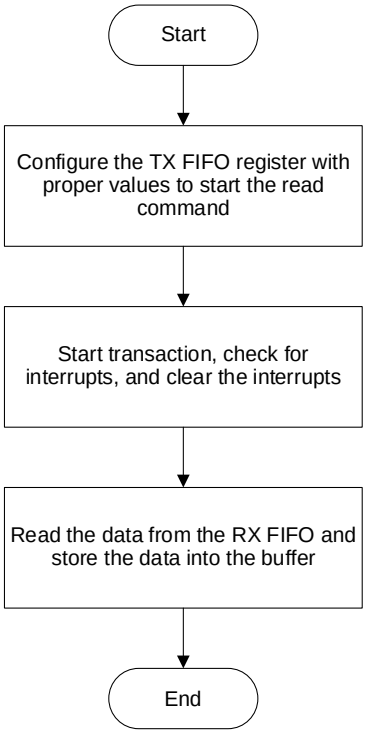

Figure 3.9. void qspi_flash_cntl_read_fifo_en()

3.10. qspi_flash_cntl_read_fifo_dis()

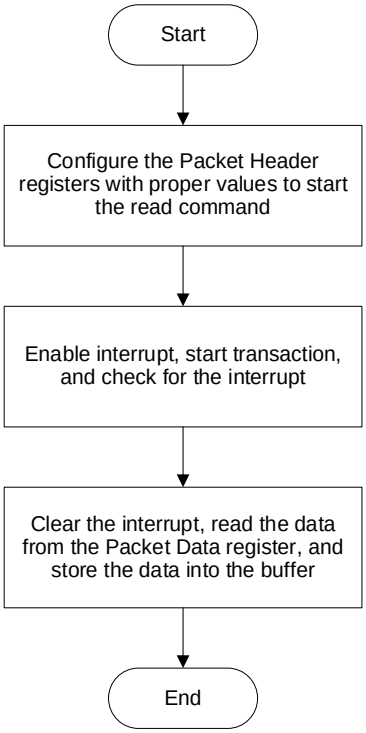

Figure 3.10. unsigned char qspi_flash_cntl_read_fifo_dis()

3.11. qspi_flash_cntl_set_pkt_hdr()

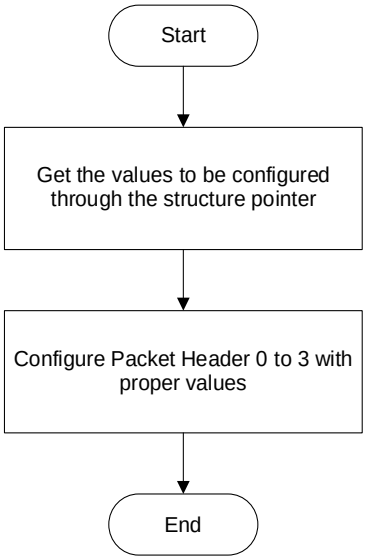

Figure 3.11. void qspi_flash_cntl_set_pkt_hdr()

3.12. qspi_flash_cntl_set_tx_fifo()

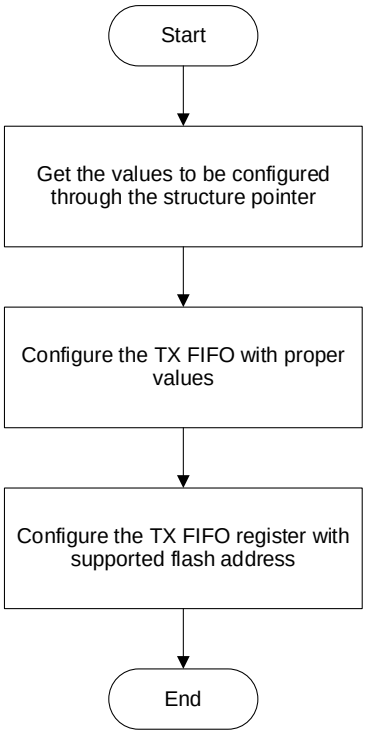

Figure 3.12. void qspi_flash_cntl_set_tx_fifo()

3.13. quad_mode()

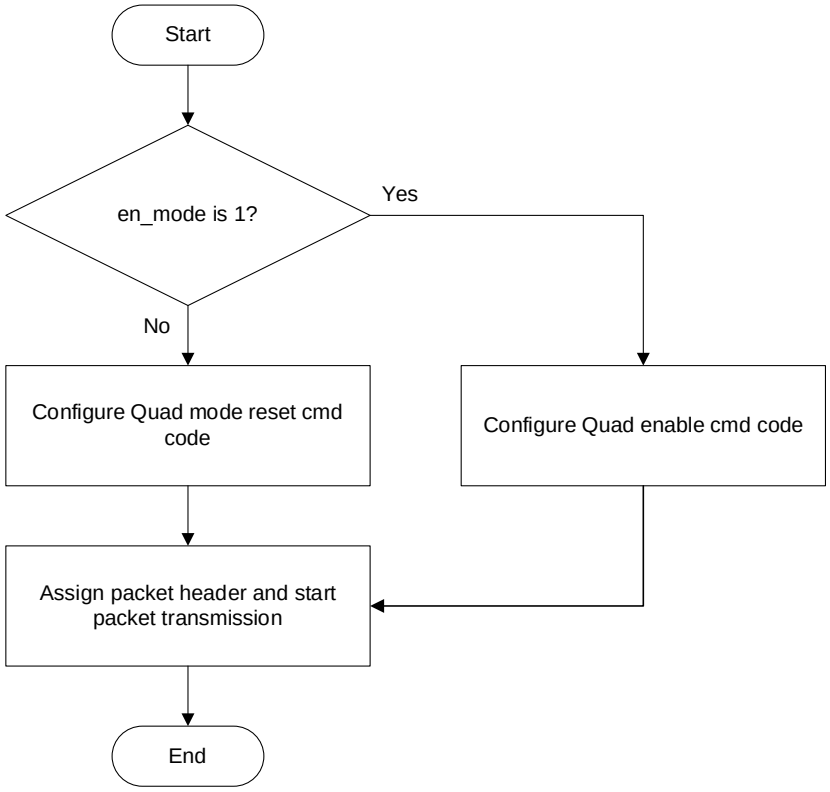

Figure 3.13. unsigned char quad_mode()

3.14. latch_set_clear()

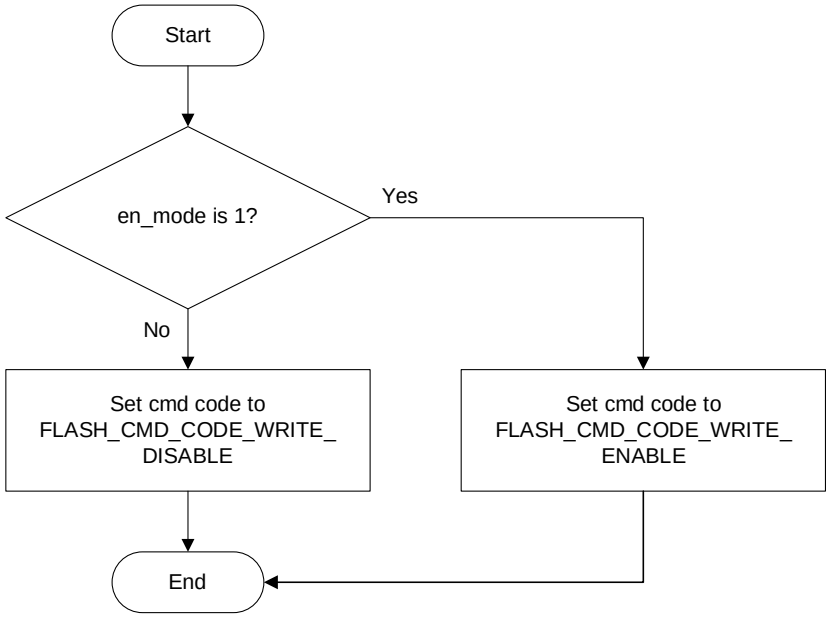

Figure 3.14. unsigned char latch_set_clear()

3.15. four_byte_mode

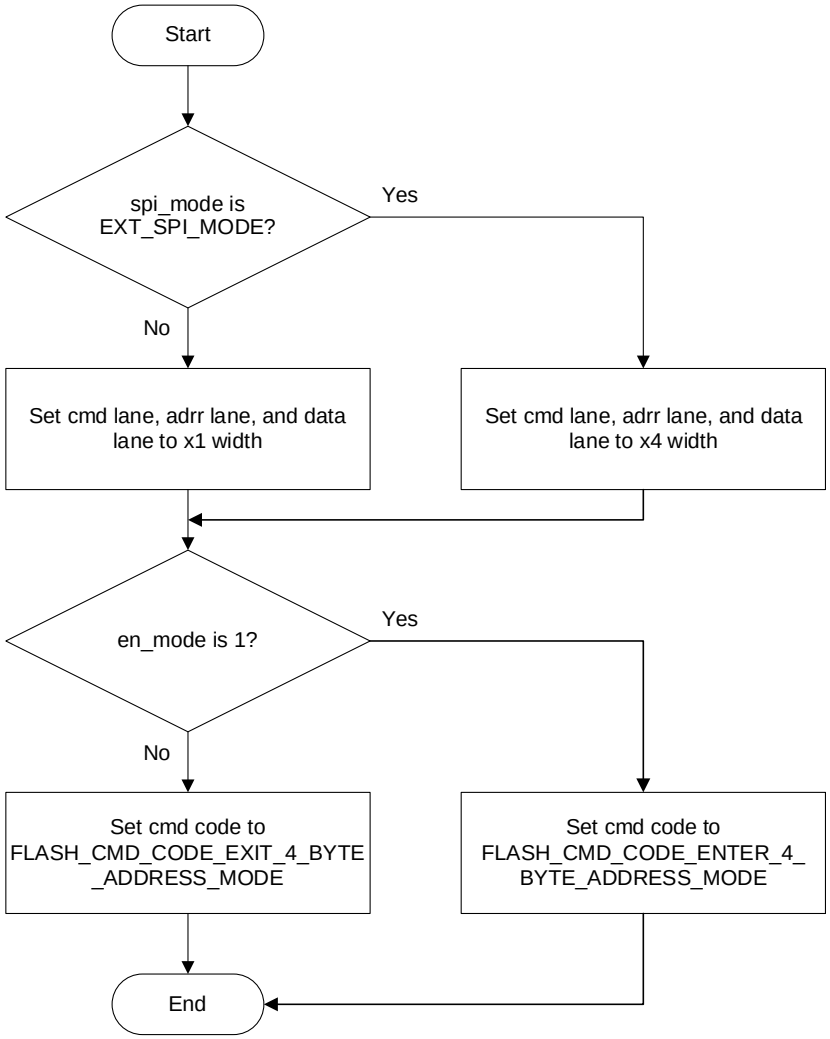

Figure 3.15. unsigned char four_byte_mode()

3.16. qspi_flash_cntl_write_config_reg()

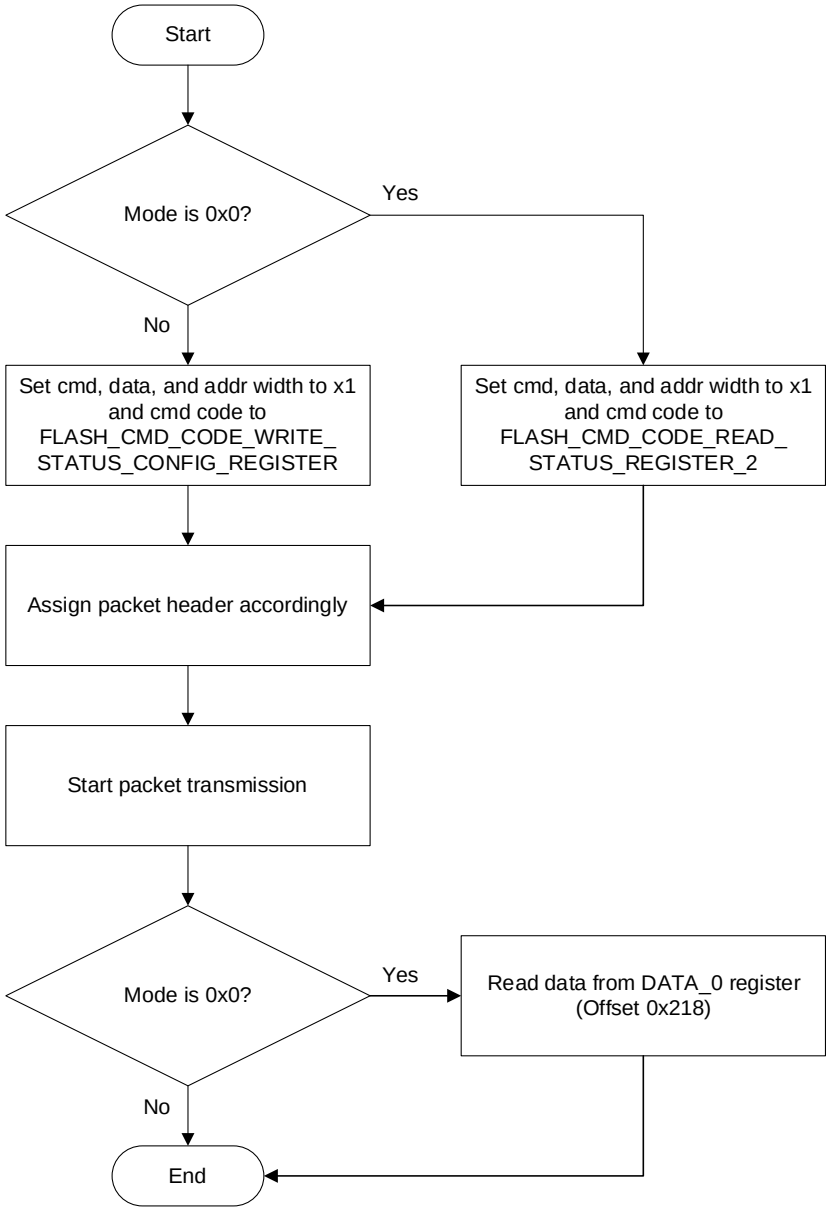

Figure 3.16. void qspi_flash_cntl_write_config_reg()

4. API Data Structures

4.1. struct qspi_flash_cntl_instance

Table 4.1. qspi_flash_cntl_instance Parameters

Data Type	Struct Member	Description
unsigned int	base_address	Base address of the flash controller.
unsigned int	status	Status of the controller.
unsigned int	config_0	Config 0 register
unsigned int	config_1	Config 1 register

4.2. struct qspi_flash_cntl_reg_t

Table 4.2. qspi_flash_cntl_reg_t Parameters

Data Type	Struct Member	Description
volatile unsigned int	REG_RESERVED	Reserved for configuration register.
volatile unsigned int	REG_QSPI_CONFIG_0	Programmable register to configure Standard/Dual/Quad SPI transactions.
volatile unsigned int	REG_QSPI_FLASH_CMD_CODE_0	Supported flash read command.
volatile unsigned int	REG_QSPI_FLASH_CMD_CODE_1	Supported flash read command.
volatile unsigned int	REG_QSPI_FLASH_CMD_CODE_2	Supported flash read command.
volatile unsigned int	REG_QSPI_FLASH_CMD_CODE_3	Supported flash read command.
volatile unsigned int	REG_QSPI_FLASH_CMD_CODE_4	Supported flash read command.
volatile unsigned int	REG_QSPI_FLASH_CMD_CODE_5	Supported flash read command.
volatile unsigned int	REG_QSPI_FLASH_CMD_CODE_6	Supported flash read command.
volatile unsigned int	REG_QSPI_FLASH_CMD_CODE_7	Supported flash read command.
volatile unsigned int	REG_MIN_FLASH_ADDR_ALIGN	Flash address alignment size for multiple target devices.
volatile unsigned int	REG_START_FLASH_ADDR	Starting actual flash memory address.
volatile unsigned int	REG_FLASH_MEM_MAP_SIZE	Size of the virtual flash memory space.
volatile unsigned int	REG_AXI_ADDR_MAP	AXI Address Map for flash memory.
volatile unsigned int	reserved_reg1[49]	Reserved addresses for configuration registers.
volatile unsigned int	REG_TRANS_STATUS	Indicates the IP transaction status.
volatile unsigned int	REG_INT_STATUS	Interrupt capability.
volatile unsigned int	REG_INT_ENABLE	Interrupt capability.
volatile unsigned int	REG_INT_SET	Interrupt capability.
volatile unsigned int	REG_SUPP_FLASH_CMD_PKT_HDR	Supported flash command packet transfer.
volatile unsigned int	reserved_reg2[58]	Reserved addresses for status and interrupt registers.
volatile unsigned int	REG_TX_FIFO_MAP	Transmit FIFO Mapping.
volatile unsigned int	REG_RX_FIFO_MAP	Receive FIFO Mapping.
volatile unsigned int	REG_PACKET_HDR_0	User Packet structure for supported and unsupported flash commands.
volatile unsigned int	REG_PACKET_HDR_1	User Packet structure for supported flash commands.
volatile unsigned int	REG_PACKET_HDR_2	User Packet structure for supported flash commands.
volatile unsigned int	REG_PACKET_HDR_3	User Packet structure for supported flash commands.
volatile unsigned int	REG_PACKET_DATA_0	User Packet structure for supported and unsupported flash commands.
volatile unsigned int	REG_PACKET_DATA_1	User Packet structure for supported and unsupported flash commands.

Data Type	Struct Member	Description
volatile unsigned int	REG_START_TRANS	Start IP operation to generate SPI transaction.

4.3. struct qspi_params_ext_t

Table 4.3. qspi_params_ext_t Parameters

Data Type	Struct Member	Bit Width	Description
unsigned int	supp_flash_cmd	1	Indicates if flash command is supported by the IP.
unsigned int	with_payload	1	Indicates with or without payload.
unsigned int	flash_cmd_code	5	Flash commands supported by the IP.
unsigned int	mult_flash_target	1	Multiple flash targets.
unsigned int	num_wait_cycle	8	Indicates the number of wait cycles.
unsigned int	buff_length	16	Length of the buffer.
unsigned int	cmd_lane_width	2	Command lane width.
unsigned int	addr_lane_width	2	Address lane width.
unsigned int	data_lane_width	2	Data lane width.
unsigned int	reservedbits_2	2	Reserved bits.
unsigned int	tgt_cs	5	Target chip select.
unsigned int	flash_addr_width	3	Flash address width to be used for the transaction.
unsigned int	reservedbits_1	16	Reserved bits.
unsigned int	flash_addr_LSB	32	Flash address in LSB mode.
unsigned int	flash_addr_MSB	32	Flash address in MSB mode.

4.4. struct qspi_params_unsupp_t

Table 4.4. qspi_params_unspp_t Parameters

Data Type	Struct Member	Bit Width	Description
unsigned int	supp_flash_cmd	1	Indicates if flash command is supported by the IP.
unsigned int	cmd_type	1	Command type to perform transaction.
unsigned int	lane_width	2	Lane width of the SPI.
unsigned int	data_rate	1	SPI data rate.
unsigned int	frm_start	1	Indicates the start of frame.
unsigned int	frm_end	1	Indicates the end of frame.
unsigned int	multi_tgt	1	Indicates multiple targets.
unsigned int	tgt_cs	5	Target CS pin.
unsigned int	num_wait_sck	3	Indicates the number of wait cycles or dummy cycles after the flash address is transmitted during read commands.
unsigned int	xfer_len_bytes	16	Transfer length in bytes. The total number of bytes goes up to 65535 bytes (64 KB).
unsigned int	header_1_reserved	32	Reserved bits.
unsigned int	header_2_reserved	32	Reserved bits.
unsigned int	header_3_reserved	32	Reserved bits.

4.5. struct qspi_params_reg_t

Table 4.4. qspi_params_reg_t Parameters

Data Type	Struct Member	Description
unsigned int	packet_header0	Packet structure for supported and unsupported flash commands.
unsigned int	packet_header1	Packet structure for supported and flash commands.
unsigned int	packet_header2	Packet structure for supported and flash commands.
unsigned int	packet_header3	Packet structure for supported and flash commands.

5. Union Data Structures

5.1. union qspi_params_t

Table 5.1. qspi_params_t Parameters

Data Type	Union Member	Description
qspi_params_ext_t	params_ext_t	Variable to access the qspi_params_ext_t struct members.
qspi_params_reg_t	params_reg_t	Variable to access the qspi_params_reg_t struct members.

6. API Enum

6.1. enum qspi_reg_type_t

Table 6.1. qspi_reg_type_t Variables

Enum Member	Enum Decimal Value
REG_QSPI_CONFIG_0	1
REG_QSPI_CONFIG_0	2
REG_QSPI_FLASH_CMD_CODE_0	3
REG_QSPI_FLASH_CMD_CODE_1	4
REG_QSPI_FLASH_CMD_CODE_2	5
REG_QSPI_FLASH_CMD_CODE_3	6
REG_QSPI_FLASH_CMD_CODE_4	7
REG_QSPI_FLASH_CMD_CODE_5	8
REG_QSPI_FLASH_CMD_CODE_6	9
REG_QSPI_FLASH_CMD_CODE_7	10
REG_MIN_FLASH_ADDR_ALIGN	11
REG_START_FLASH_ADDR	12
REG_FLASH_MEM_MAP_SIZE	13
REG_AXI_ADDR_MAP	14
REG_TRANS_STATUS	64
REG_INT_STATUS	65
REG_INT_ENABLE	66
REG_INT_SET	67
REG_SUPP_FLASH_CMD_PKT_HDR	68
REG_UNSUPP_FLASH_CMD_PKT_HDR	69
REG_TX_FIFO_MAP	128
REG_RX_FIFO_MAP	129
REG_PACKET_HDR_0	130
REG_PACKET_HDR_1	131
REG_PACKET_HDR_2	132
REG_PACKET_HDR_3	133
REG_PACKET_DATA_0	134
REG_PACKET_DATA_1	135
REG_START_TRANS	136
REG_EN_LOOPBACK_TEST	137

6.2. enum lane_width_t

Table 6.2. lane_width_t Variables

Enum Member	Value
LANE_WIDTH_X1	0x0
LANE_WIDTH_X2	0x1
LANE_WIDTH_X3	0x2

6.3. enum flash_addr_width_t

Table 6.3. flash_addr_width_t Variables

Enum Member	Value
FLASH_NO_ADDR	0x0
FLASH_16BIT_ADDR	0x1
FLASH_24BIT_ADDR	0x2
FLASH_32BIT_ADDR	0x3
FLASH_40BIT_ADDR	0x4
FLASH_48BIT_ADDR	0x5
FLASH_56BIT_ADDR	0x6
FLASH_64BIT_ADDR	0x7

6.4. enum flash_cmd_code_table_t

Table 6.4. flash_cmd_code_table_t Variables

Enum Member	Value
FLASH_CMD_CODE_READ_STATUS_REGISTER_1	0x0
FLASH_CMD_CODE_READ_STATUS_REGISTER_2	0x1
FLASH_CMD_CODE_READ_STATUS_REGISTER_3	0x2
FLASH_CMD_CODE_READ_CONFIGURATION_REGISTER	0x3
FLASH_CMD_CODE_READ_ID	0x4
FLASH_CMD_CODE_READ_ELECTRONIC_ID	0x5
FLASH_CMD_CODE_MULTIPLE_IO_READ_ID	0x6
FLASH_CMD_CODE_READ_MANUFACTURER_AND_DEVICE_ID	0x7
FLASH_CMD_CODE_READ_MANUFACTURER_AND_DEVICE_ID_DUAL	0x8
FLASH_CMD_CODE_READ_MANUFACTURER_AND_DEVICE_ID_QUAD	0x9
FLASH_CMD_CODE_READ_DATA	0xA
FLASH_CMD_CODE_FAST_READ	0xB
FLASH_CMD_CODE_DUAL_OUTPUT_FAST_READ	0xC
FLASH_CMD_CODE_DUAL_INPUT_OUTPUT_FAST_READ	0xD
FLASH_CMD_CODE_QUAD_OUTPUT_FAST_READ	0xE
FLASH_CMD_CODE_QUAD_INPUT_OUTPUT_FAST_READ	0xF
FLASH_CMD_CODE_BLOCK_ERASE_TYPE_1	0x10
FLASH_CMD_CODE_BLOCK_ERASE_TYPE_2	0x11
FLASH_CMD_CODE_BLOCK_ERASE_TYPE_3	0x12
FLASH_CMD_CODE_CHIP_ERASE	0x13
FLASH_CMD_CODE_WRITE_ENABLE	0x14
FLASH_CMD_CODE_WRITE_DISABLE	0x15
FLASH_CMD_CODE_WRITE_STATUS_CONFIG_REGISTER	0x16
FLASH_CMD_CODE_PAGE_PROGRAM	0x17
FLASH_CMD_CODE_DUAL_INPUT_FAST_PROGRAM	0x18
FLASH_CMD_CODE_EXTENDED_DUAL_INPUT_FAST_PROGRAM	0x19
FLASH_CMD_CODE_QUAD_INPUT_FAST_PROGRAM	0x1A
FLASH_CMD_CODE_EXTENDED_QUAD_INPUT_FAST_PROGRAM	0x1B
FLASH_CMD_CODE_ENTER_4_BYTE_ADDRESS_MODE	0x1C
FLASH_CMD_CODE_EXIT_4_BYTE_ADDRESS_MODE	0x1D
FLASH_CMD_CODE_ENTER_QUAD_INPUT_OUTPUT_MODE	0x1E

Enum Member	Value
FLASH_CMD_CODE_EXIT_QUAD_INPUT_OUTPUT_MODE	0x1F

6.5. enum config_output_type_t

Table 6.5. enum config_output_type_t Variables

Enum Member	Value
PRINT_ALL	0x0
PRINT_ONLY_CONFIG	0x1
PRINT_ONLY_HDR_DATA	0x2

6.6. enum spi_mode_t

Table 6.6. enum spi_mode_t Variables

Enum Member	Value
FLASH_CNTL_SPI_MODE	0x0
FLASH_CNTL_QSPI_MODE	0x1
FLASH_CNTL_EXTENDED_SPI_MODE	0x2

6.7. enum erase_type_t

Table 6.7. enum erase_type_t Variables

Enum Member	Value
ERASE_TYPE_4K	0x0
ERASE_TYPE_32K	0x1
ERASE_TYPE_64K	0x2
ERASE_TYPE_CHIP	0x3

7. API Variable

```
qspi_flash_cntl_reg_t  *flash_cntl
```

This variable is used to access the data members (registers) of the given structure.

8. API Macros

Table 8.1. API Macros Description

Macro	Description
#define QSPI_FLASH_CONTROLLER_DRV_VER	QSPI Driver version.
#define CLR_INTERRUPT	Used to clear the interrupts.
#define FLASH_ADDRESS_FIFO_DIS	Flash address when FIFO is disabled.
#define FLASH_ADDRESS_FIFO_EN	Flash address when FIFO is enabled.
#define START_TRANS_VAL	Transaction value to start the transaction.
#define CLR_START_TRANS_VAL	Clear transaction.
#define FLASH_CMD_CODE_READ	Flash command code for read.
#define FLASH_CMD_CODE_WRITE	Flash command code for write.
#define NUM_WAIT_CYCLE	Indicates the number of wait cycles.
#define MULT_FLASH_TGT	Indicates the number of flash targets.
#define WITH_PAYLOAD_WR	Indicates payload is present.
#define WITH_PAYLOAD_RD	Indicates payload is not present.
#define SUPP_FLASH_CMD	Indicates if flash command is supported the IP.
#define FLASH_ADDR_WIDTH_FIFO_DIS	Indicates flash address width when FIFO is disabled.
#define TGT_CS	Target chip select.
#define ADDR_LANE_WIDTH	Address lane width sent to target flash memory.
#define CMD_LANE_WIDTH	Command lane width sent to target flash memory.
#define FLASH_ADDR_WIDTH_FIFO_EN	Indicates flash address width when FIFO is enabled.
#define RET_FAILURE	Return failure: 1.
#define RET_SUCCESS	Return success: 0.
#define IDLE	Indicates the idle state value.
#define IS_NULL	Checks the null condition.
#define TRANSFER_LEN_SHIFT	Shifts the transfer length.
#define FLASH_ADDR_WID_SHIFT_FIFO_EN	Flash address shifting when FIFO is enabled.
#define FLASH_ADDR_WID_SHIFT_FIFO_DIS	Flash address shifting when FIFO is disabled.
#define FLASH_CMD_CODE_SHIFT	Shifts the flash command code.
#define NUM_WAIT_CYCLE_SHIFT	Shifts the number of wait cycles.
#define MULT_FLASH_TGT_SHIFT	Shifts the multiple flash targets.
#define WITH_PAYLOAD_SHIFT	Shifts the payload value.
#define TGT_CS_SHIFT	Shifts the target chip select value.
#define DATA_LANE_WIDTH_SHIFT	Shifts the data lane width value.
#define ADDR_LANE_WIDTH_SHIFT	Shifts the address lane width value.
#define INT_EN_VAL	Value for enabling the interrupt.
#define SET_PKT_HDR3	Sets packet header 3 value.
#define CLR_START_TRANS	Clears start transaction.
#define REG_GAP	Register gap value.
#define IRQ_WR_FIFO_EN	Interrupt write when FIFO is enabled.
#define IRQ_RD_FIFO_EN	Interrupt read when FIFO is enabled.
#define IRQ_WR_FIFO_DIS	Interrupt write when FIFO is disabled.
#define IRQ_RD_FIFO_DIS	Interrupt read when FIFO is disabled.
#define TRANS_STS	Transaction status.
#define CLR_IRQ_STS_VAR	Clear interrupt status variable.
#define IDX_ZERO	Index zero.
#define IDX_ONE	Index one.

Macro	Description
#define IDX_TWO	Index two.
#define INCREMENT_ONE	Increment one.
#define INCREMENT_TWO	Increment two.
#define ENABLE_XIP_TRANS_BIT	XiP enable bit in Transaction register
#define DISABLE_XIP_TRANS_BIT	XiP disable bit in Transaction register
#define ENABLE	Enable bit
#define DISABLE	Disable bit
#define TGT_RD_TRANS_CNT	Target read transaction count.
#define TGT_WR_TRANS_CNT	Target write transaction count.
#define QUAD_IO_FAST_RD_CMD_CODE	Quad input output fast read command.
#define ENTER_4_BYTE_ADDR_MODE_CMD_CODE	Enter 4-byte address mode command.
#define EXIT_4_BYTE_ADDR_MODE_CMD_CODE	Exit 4-byte address mode command.
#define ENTER_QUAD_IO_MODE_CMD_CODE	Enter quad input output mode command.
#define RESET_QUAD_IO_CMD_CODE	Reset quad input output command.
#define FLASH_CMD_CODE_7_REG_CMD_QUAD_EN	Quad enable command.
#define PKT_HDR_0_CMD_QUAD_EN	Quad enable for Packet Header 0.
#define PKT_HDR_1_CMD_QUAD_EN	Quad enable for Packet Header 1.
#define FLASH_CMD_CODE_7_REG_CMD_QUAD_RST	Quad reset command.
#define PKT_HDR_0_CMD_QUAD_RST	Quad reset for Packet Header 0.
#define PKT_HDR_1_CMD_QUAD_RST	Quad reset for Packet Header 1.
#define PKT_HDR_0_XIP_ENABLE	XiP enable for Packet Header 0
#define PKT_HDR_0_XIP_ENABLE_READ	XiP enable read back for Packet Header 0
#define PKT_DATA_0_XIP_ENABLE	XiP enable for Packet Data 0
#define PKT_HDR_0_XIP_DISABLE	XiP disable for Packet Header 0
#define PKT_DATA_0_XIP_DISABLE	XiP disable for Packet Data 0
#define PKT_HDR_0_XIP_DISABLE_READ	XiP disable read back for Packet Header 0
#define CHECK_ONE	Checks for one.
#define CHECK_TWO	Checks for two
#define SHIFT_FOUR	Shifts by four.
#define FLASH_ADDR_SHIFT	Shifts flash address.
#define FLASH_ADDR_MSB	Flash Address MSB.
#define EN_FRAME_END_DONE_CNTR	Enables frame end done counter.
#define LITTLE_ENDIAN	Little Endian data.
#define BIG_ENDIAN	Big Endian data.
#define USE_BIG_ENDIAN	Big endian to use for SPI transaction
#define DIV_BY_2	Division by 2 clock rate.
#define DIV_BY_4	Division by 4 clock rate.
#define DIV_BY_8	Division by 8 clock rate.
#define SAMPLE_ODD_EDGES	Sample at odd edges clock phase.
#define SAMPLE_EVEN_EDGES	Sample at even edges clock phase.
#define ACTIVE_LOW	Active low clock polarity.
#define ACTIVE_HIGH	Active high clock polarity.
#define MSB_FIRST	MSB first.
#define LSB_FIRST	LSB first.
#define XFER_LEN_BYTES	Transfer Length.
#define NUM_WAIT_STATE	Indicates the number of wait states.
#define MULT_FLASH_TARGET	Indicates the number of flash targets.

Macro	Description
#define FLASH_CMD_CODE	Indicates flash command code.
#define WITH_PAYLOAD	Indicates payload.
#define SUPPORTED_FLASH_CMD	Indicates if flash command is supported by the IP.
#define FLASH_ADDRESS_WIDTH	Indicates flash address width.
#define TARGET_CS	Target chip select.
#define DATA_LANE_WIDTH	Data lane width sent to target flash memory.
#define ADDRESS_LANE_WIDTH	Address lane width sent to target flash memory.
#define COMMAND_LANE_WIDTH	Command lane width sent to target flash memory.
#define JEDEC_MANUFACTURER_ID_MICRON	Manufacturer ID of Micron.
#define JEDEC_MANUFACTURER_ID_MACRONIX	Manufacturer ID of Macronix.
#define JEDEC_MANUFACTURER_ID_WINBOND	Manufacturer ID of Winbond.
#define OPCODE_3BYTE_BLOCK_ERASE_1	3 bytes block erase command 1.
#define OPCODE_3BYTE_BLOCK_ERASE_2	3 bytes block erase command 2.
#define OPCODE_3BYTE_BLOCK_ERASE_3	3 bytes block erase command 3.
#define OPCODE_4BYTE_BLOCK_ERASE_1	4 bytes block erase command 1.
#define OPCODE_4BYTE_BLOCK_ERASE_2	4 bytes block erase command 2.
#define OPCODE_4BYTE_BLOCK_ERASE_3	4 bytes block erase command 3.
#define INT_REG_RD_PKT_DATA_1_NOT_EMPTY	Indicates read data packet DATA_1 register is not empty.
#define INT_REG_RD_PKT_DATA_0_NOT_EMPTY	Indicates read data packet DATA_0 register is not empty.
#define INT_REG_WR_PKT_DATA_1_EMPTY	Indicates write data packet DATA_1 register is not empty.
#define INT_REG_WR_PKT_DATA_0_EMPTY	Indicates write data packet DATA_0 register is not empty.
#define WINBOND_QSPI_QE_EN_BIT	Winbond QE bit enable.
#define WRITE_EN_CMD_CODE	Write enable command code.
#define WRITE_DIS_CMD_CODE	Write disable command code.
#define WRITE_STS_CONFIG_REG_CMD_CODE	Write status config register command code.
#define PAGE_PROG_CMD_CODE	Page program command code.

References

- QSPI Flash Controller IP User Guide (FPGA-IPUG-02248)
- QSPI Flash Controller IP Release Notes (FPGA-RN-02016)
- Avant-E web page
- CertusPro-NX web page
- QSPI Flash Controller IP Core web page
- Lattice Radiant Software web page
- Lattice Propel Design Environment web page
- Lattice Solutions IP Cores web page
- Lattice Solutions Reference Designs web page
- Lattice Insights for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 1.4, June 2025

Section	Change Summary
Abbreviations in This Document	Added *WEL*.

Introduction	• Made minor editorial change to the IP user guide reference.
	• In the Driver Version section:
	  • Updated section title and removed subsection.
	  • Updated driver version.
	• In the Driver and IP Compatibility section:
	  • Updated driver version and IP version.
	  • Updated tool versions.
	  • Added reference to release notes.

API Description	• Updated function description in the qspi_flash_cntl_write_fifo_dis() section.
	• In the latch_set_clear() section:
	  • Updated function description.
	  • Updated en_mode parameter description.
	• In the four_byte_mode() section:
	  • Updated en_mode parameter description.

References	• Added QSPI Flash Controller IP Release Notes (FPGA-RN-02016). • Added QSPI Flash Controller IP Core, Lattice Propel Design Environment, and Lattice Solutions Reference Designs web pages. • Updated listing name to Lattice Radiant Software web page.

Revision 1.3, January 2025

Section	Change Summary
Introduction	• Updated driver version in 1.3.1. Driver Version section. • Updated driver and IP versions in 1.4. Driver and IP Compatibility section.
API Description	• Added 2.1. print_config_reg(), 2.2. qspi_trans_start(), 2.6. qspi_flash_cntl_read_id(), 2.7. qspi_flash_cntl_erase(), 2.14. latch_set_clear(), 2.15. four_byte_mode(), and 2.16. qspi_flash_cntl_read_write_config_reg() sections. • Updated function in 2.5. qspi_flash_cntl_write_fifo_dis() section. • Updated function in 2.13. quad_mode() section. • Removed qspi_flash_cntl_write_erase_fifo_dis(), reset_quad_mode(), enable_xip_mode(), and xip_read_data() sections.
Function Call Flow Diagrams	• Added 3.1. print_config_reg(), 3.2. qspi_trans_start(), 3.6. qspi_flash_cnt_read_id(), 3.7. qspi_flash_cntl_erase(), 3.14. latch_set_clear(), 3.15. four_byte_mode, and 3.16. qspi_flash_cntl_write_config_reg() sections. • Updated Figure 3.5. unsigned char qspi_flash_cntl_write_fifo_dis() in 3.5. qspi_flash_cntl_write_fifo_dis() section. • Updated Figure 3.13. unsigned char quad_mode() in 3.13. quad_mode() section. • Removed qspi_flash_cntl_write_erase_fifo_dis(), reset_quad_mode(), enable_xip_mode(), and xip_read_data() sections.
API Data Structures	Added 4.4. struct qspi_params_unsupp_t section.
API Enum	Added 6.5. enum config_output_type_t, 6.6. enum spi_mode_t, and 6.7. enum erase_type_t sections.
API Macros	In Table 8.1. API Macros Description: • Added macros #define OPCODE_3BYTE_BLOCK_ERASE_1/2/3, #define OPCODE_4BYTE_BLOCK_ERASE_1/2/3, #define INT_REG_RD_PKT_DATA_1_NOT_EMPTY, #define INT_REG_RD_PKT_DATA_0_NOT_EMPTY #define INT_REG_WR_PKT_DATA_1_EMPTY, #define INT_REG_WR_PKT_DATA_0_EMPTY, #define WINBOND_QPSI_QE_EN_BIT, #define WRITE_EN_CMD_CODE, #define WRITE_DIS_CMD_CODE, #define

Section	Change Summary
	WRITE_STS_CONFIG_REG_CMD_CODE, and #define PAGE_PROG_CMD_CODE.

Revision 1.2, August 2024

Section	Change Summary
Abbreviations in This Document	Minor editorial changes.

Introduction	• Updated description in Audience section.
	• Updated driver version in 1.3.1 Driver Version.
	• In 1.4 Driver and IP Compatibility section:
	  • Updated driver and IP versions.
	  • Added table for *Driver tested on hardware device*.

API Description	• Removed *unsigned int base_addr* and *base_addr* from qspi_flash_cntl_init() input parameters in the 2.1 qspi_flash_cntl_init() section. • Updated function return type in 2.6 qspi_flash_cntl_write_erase_fifo_dis() section. • Updated function return type, added function return value, and updated function parameters in 2.7 qspi_flash_cntl_read_fifo_dis() section. • Added table in 2.10 enable_quad_mode() and 2.11 reset_quad_mode() sections. • Added 2.12 enable_xip_mode() and 2.13 xip_read_data() sections.
Function Call Flow Diagrams	• Updated Figure 3.6. unsigned char qspi_flash_cntl_write_erase_fifo_dis() and figure title in 3.6 qspi_flash_cntl_write_erase_fifo_dis() section. • Updated Figure 3.7. unsigned char qspi_flash_cntl_read_fifo_dis() figure title in 3.7 qspi_flash_cntl_read_fifo_dis() section. • Updated Figure 3.8. void qspi_flash_cntl_set_pkt_hdr() in 3.8 qspi_flash_cntl_set_pkt_hdr() section. • Added 3.12 enable_xip_mode() and 3.13 xip_read_data() sections.
API Data Structures	Removed structs REG_UNSUPP_FLASH_CMD_PKT_HDR and REG_EN_LOOPBACK_TEST from Table 4.2. qspi_flash_cntl_reg_t Parameters in the struct qspi_flash_cntl_reg_t section.
API Enum	Added enum flash_cmd_code_table_t section.
API Macros	In Table 8.1. API Macros Description: • Removed macros #define EN_FLASH_ADDR_SPACE_MAP, #define CHIP_SEL_BEH, #define MIN_IDLE_TIME, and #define EN_BACK_TO_BACK_TRANS. • Added macros #define ENABLE_XIP_TRANS_BIT, #define DISABLE_XIP_TRANS_BIT, #define ENABLE, #define DISABLE, #define PKT_HDR_0_XIP_ENABLE, #define PKT_HDR_0_XIP_ENABLE_READ, #define PKT_DATA_0_XIP_ENABLE, #define PKT_HDR_0_XIP_DISABLE, #define PKT_DATA_0_XIP_DISABLE, #define PKT_HDR_0_XIP_DISABLE_READ, #define CHECK_TWO, #define USE_BIG_ENDIAN, #define JEDEC_MANUFACTURER_ID_MICRON, #define JEDEC_MANUFACTURER_ID_MACRONIX, and #define JEDEC_MANUFACTURER_ID_WINBOND.

Revision 1.1, February 2024

Section	Change Summary
Inclusive Language	Added boilerplate.
Introduction	• Updated the naming convention of driver in 1.3.1. Driver Version section. • Updated the IP version from *1.1.0* to 1.1.1 in 1.4. Driver and IP Compatibility section.
API Description	• Removed the *en_quad_io_mode_cmd* parameter in 2.10. enable_quad_mode() section. • Removed the *rst_quad_io_mode_cmd* parameter in 2.11. reset_quad_mode() section.
Function Call Flow Diagrams	Updated the section headers for 3.1. qspi_flash_cntl_init() – 3.11. reset_quad_mode() sections.
API Data Structures	• Added Struct Members *config_0* and *config_1* in Table 4.1. qspi_flash_cntl_instance Parameters. • Removed one of the *REG_QSPI_CONFIG_0* Struct Members due to duplication in Table 4.2.

Section	Change Summary
	qspi_flash_cntl_reg_t Parameters. • Added Struct Member *REG_RESERVED* in Table 4.2. qspi_flash_cntl_reg_t Parameters. • Added *ext* in section header of 4.3. struct qspi_params_ext_t section and table caption of Table 4.3. qspi_params_ext_t Parameters. • Added *Bit Width* column to Table 4.3. qspi_params_ext_t Parameters. • Removed Struct Member *flash_addr* from Table 4.3. qspi_params_ext_t Parameters. • Added Struct Members *reservedbits_1*, *reservedbits_2*, *flash_addr_LSB*, and *flash_addr_MSB* to Table 4.3. qspi_params_ext_t Parameters. • Added 4.4. struct qspi_params_reg_t section.
Union Data Structures	Added this section.
API Enum	• Updated numberings for section headers and table captions. • Added 6.2. enum lane_width_t and 6.3. enum flash_addr_width_t sections.
API Variable	Updated numbering for section header.
API Macros	• Updated numberings for section headers and table captions. • Removed macros: *#define CPHA*, *#define CPOL*, *#define DATA_ENDIANNESS*, *#define FAILURE*, *#define FIRST_BIT_TRANS*, *#define FLASH_ADDR_WIDTH*, *#define FLASH_ADDR_WIDTH*, *#define FLASH_CMD_CODE_&_REG_CMD_QUAD_EN*, *#define FLASH_CMD_CODE_&_REG_CMD_QUAD_RST*, *#define QUAD_IO_MODE_SHIFT*, *#define SCK_RATE*, *#define SUCCESS*, and *#define TRANSACTION_VAL* from Table 8.1. API Macros Description. • Added macros: *#define ACTIVE_HIGH*, *#define ACTIVE_LOW*, *#define ADDRESS_LANE_WIDTH*, *#define BIG_ENDIAN*, *#define CLR_START_TRANS_VAL*, *#define COMMAND_LANE_WIDTH*, *#define DIV_BY_2*, *#define DIV_BY_4*, *#define DIV_BY_8*, *#define ENTER_4_BYTE_ADDR_MODE_CMD_CODE*, *#define ENTER_QUAD_IO_MODE_CMD_CODE*, *#define EXIT_4_BYTE_ADDR_MODE_CMD_CODE*, *#define FLASH_ADDR_MSB*, *#define FLASH_ADDR_SHIFT*, *#define FLASH_ADDR_WIDTH_FIFO_DIS*, *#define FLASH_ADDR_WIDTH_FIFO_EN*, *#define FLASH_ADDRESS_WIDTH*, *#define FLASH_CMD_CODE*, *#define FLASH_CMD_CODE_7_REG_CMD_QUAD_EN*, *#define FLASH_CMD_CODE_7_REG_CMD_QUAD_RST*, *#define LITTLE_ENDIAN*, *#define LSB_FIRST*, *#define MSB_FIRST*, *#define MULT_FLASH_TARGET*, *#define NUM_WAIT_STATE*, *#define QSPI_FLASH_CONTROLLER_DRV_VER*, *#define QUAD_IO_FAST_RD_CMD_CODE*, *#define RESET_QUAD_IO_CMD_CODE*, *#define RET_FAILURE*, *#define RET_SUCCESS*, *#define SAMPLE_EVEN_EDGES*, *#define SAMPLE_ODD_EDGES*, *#define START_TRANS_VAL*, *#define SUPPORTED_FLASH_CMD*, *#define TARGET_CS*, *#define WITH_PAYLOAD*, and *#define XFER_LEN_BYTES* to Table 8.1. API Macros Description.

Revision 1.0, December 2023

Section	Change Summary
All	Initial release.

www.latticesemi.com