xSPI Target Driver API Reference

Technical Note

FPGA-TN-02426-1.0

June 2026

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
ASCII	American Standard Code for Information Interchange
CPHA	Clock Phase
CPOL	Clock Polarity
CSR	Configuration Status Register
FIFO	First In, First Out
FPGA	Field Programmable Gate Array
I/O	Input/Output
ID	Identification
IP	Intellectual Property
LSB	Least Significant Bit
MSB	Most Significant Bit
Rx	Receiver
SPI	Serial Peripheral Interface
Tx	Transmitter
xSPI	eXpanded Serial Peripheral Interface

1. Introduction

The Serial Peripheral Interface (SPI) is a synchronous serial communication protocol that facilitates data transfer between microcontrollers and peripheral devices. SPI is commonly used in embedded systems to interface with sensors, memory devices, and display controllers, among other peripherals. The eXpanded Serial Peripheral Interface (xSPI) is an extension of SPI that supports multiple SPI protocols. The Lattice xSPI Target IP is an SPI interface that supports standard and quad SPI protocols.

Refer to the xSPI Target IP User Guide (FPGA-IPUG-02323) for more details about the intellectual property (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 application programming interfaces (APIs) and function call flows.

1.2. Audience

The intended audience for this document includes embedded system designers and embedded software developers using Lattice CrossLink™-NX, Certus™-NX, CertusPro™-NX, MachXO5™-NX, and MachXO3D™ devices. The technical guide assumes readers have expertise in embedded systems and field programmable gate array (FPGA) technologies.

1.3. Driver Version

xSPI_TARGET_DRV_VER “26.1.0”

1.4. Driver and IP Compatibility

Driver version	IP version
26.1.0	1.0.0

Refer to the xSPI Target IP Release Notes (FPGA-RN-02115) for more information on the driver and IP versions.

2. API Description

2.1. xspi_tgt_init()

This API is used to verify that the IP unique identifier is *XSPT*, represented as 0x58535054 in American Standard Code for Information Interchange (ASCII), perform a soft reset, program all configuration registers, and finally assert EN_TARGET.

```
xspi_tgt_status_t xspi_tgt_init(xspi_tgt_dev_t *dev, uintptr_t base,
const xspi_tgt_cfg_t *cfg)
```

In/Out	Parameter	Description	Returns
In	*dev	Handle of the struct xspi_tgt_dev_t.	1: success 0: failure
In	base	Memory-mapped base address of the xSPI target register block.	
In	*cfg	Desired configuration; if NULL, the driver applies default configuration values.	

2.2. xspi_tgt_read_id()

This API is used to read the IP identification register.

```
xspi_tgt_status_t xspi_tgt_read_id(const xspi_tgt_dev_t *dev, uint32_t *out_id)
```

In/Out	Parameter	Description	Returns
In	*dev	Handle of the struct xspi_tgt_dev_t.	1: success 0: failure
Out	*out_id	Receives the 32-bit IP identification value.	

2.3. xspi_tgt_set_ready()

This API is used to set the READY and BUSY flags for SPI status auto-response.

```
xspi_tgt_status_t xspi_tgt_set_ready(xspi_tgt_dev_t *dev, bool ready)
```

In/Out	Parameter	Description	Returns
In	*dev	Handle of the struct xspi_tgt_dev_t.	1: success 0: failure
In	ready	True – READY = 1, BUSY = 0 False – READY = 0, BUSY = 1	

2.4. xspi_tgt_get_first_byte()

This API is used to read the first byte (command opcode) of the current transaction.

```
xspi_tgt_status_t xspi_tgt_get_first_byte(const xspi_tgt_dev_t *dev,
uint8_t *out_byte, bool *out_valid)
```

In/Out	Parameter	Description	Returns
In	*dev	Handle of the struct xspi_tgt_dev_t.	1: success 0: failure
Out	*out_byte	Receives the first-byte value.	
Out	*out_valid	Receives true if FIRST_BYTE flag is set (value valid).	

2.5. xspi_tgt_tx_fifo_is_full()

This API is used to check whether the transmitter (Tx) first in, first out (FIFO) is full.

```
xspi_tgt_status_t xspi_tgt_tx_fifo_is_full(const xspi_tgt_dev_t *dev, bool *out_full)
```

In/Out	Parameter	Description	Returns
In	*dev	Handle of the struct xspi_tgt_dev_t.	1: success 0: failure
Out	*out_full	Receives true if Tx FIFO is full.	

2.6. xspi_tgt_tx_fifo_is_empty()

This API is used to check whether the Tx FIFO is empty.

```
xspi_tgt_status_t xspi_tgt_tx_fifo_is_empty(const xspi_tgt_dev_t *dev, bool *out_empty)
```

In/Out	Parameter	Description	Returns
In	*dev	Handle of the struct xspi_tgt_dev_t.	1: success 0: failure
Out	*out_empty	Receives true if Tx FIFO is empty.	

2.7. xspi_tgt_rx_fifo_is_full()

This API is used to check whether the receiver (Rx) FIFO is full.

```
xspi_tgt_status_t xspi_tgt_rx_fifo_is_full(const xspi_tgt_dev_t *dev, bool *out_full)
```

In/Out	Parameter	Description	Returns
In	*dev	Handle of the struct xspi_tgt_dev_t.	1: success 0: failure
Out	*out_full	Receives true if Rx FIFO is full.	

2.8. xspi_tgt_rx_fifo_is_empty()

This API is used to check whether the Rx FIFO is empty.

```
xspi_tgt_status_t xspi_tgt_rx_fifo_is_empty(const xspi_tgt_dev_t *dev, bool *out_empty)
```

In/Out	Parameter	Description	Returns
In	*dev	Handle of the struct xspi_tgt_dev_t.	1: success 0: failure
Out	*out_empty	Receives true if Rx FIFO is empty.	

2.9. xspi_tgt_tx_byte()

This API is used to write a single byte to the Tx register or FIFO.

```
xspi_tgt_status_t xspi_tgt_tx_byte(xspi_tgt_dev_t *dev, uint8_t data)
```

In/Out	Parameter	Description	Returns
In	*dev	Handle of the struct xspi_tgt_dev_t.	1: success 0: failure
In	data	Byte to transmit.	

2.10. xspi_tgt_rx_byte()

This API is used to read a single byte from the Rx register or FIFO.

```
xspi_tgt_status_t xspi_tgt_rx_byte(const xspi_tgt_dev_t *dev, uint8_t *out_data)
```

In/Out	Parameter	Description	Returns
In	*dev	Handle of the struct xspi_tgt_dev_t.	1: success 0: failure
Out	*out_data	Receives the read byte (bits [7:0]).	

2.11. xspi_tgt_tx_write()

This API is used to write bytes from an input buffer to the Tx FIFO.

```
xspi_tgt_status_t xspi_tgt_tx_write(xspi_tgt_dev_t *dev, const uint8_t *buf, size_t len)
```

In/Out	Parameter	Description	Returns
In	*dev	Handle of the struct xspi_tgt_dev_t.	1: success 0: failure
In	*buf	Source buffer.	
In	len	Number of bytes to write.	

2.12. xspi_tgt_soft_reset_ip()

This API is used to trigger an IP core reset.

```
xspi_tgt_status_t xspi_tgt_soft_reset_ip(xspi_tgt_dev_t *dev)
```

In/Out	Parameter	Description	Returns
In	*dev	Handle of the struct xspi_tgt_dev_t.	1: success 0: failure

2.13. xspi_tgt_soft_reset_tx_fifo()

This API is used to reset the Tx FIFO only.

```
xspi_tgt_status_t xspi_tgt_soft_reset_tx_fifo(xspi_tgt_dev_t *dev)
```

In/Out	Parameter	Description	Returns
In	*dev	Handle of the struct xspi_tgt_dev_t.	1: success 0: failure

2.14. xspi_tgt_soft_reset_rx_fifo()

This API is used to reset the Rx FIFO only.

```
xspi_tgt_status_t xspi_tgt_soft_reset_rx_fifo(xspi_tgt_dev_t *dev)
```

In/Out	Parameter	Description	Returns
In	*dev	Handle of the struct xspi_tgt_dev_t.	1: success 0: failure

2.15. xspi_tgt_int_enable()

This API is used to enable selected interrupt sources.

```
xspi_tgt_status_t xspi_tgt_int_enable(xspi_tgt_dev_t *dev, uint32_t mask)
```

In/Out	Parameter	Description	Returns
In	*dev	Handle of the struct xspi_tgt_dev_t.	1: success 0: failure
In	mask	OR-ed XSPI_TGT_INT_*_MSK bits to enable.	

2.16. xspi_tgt_int_status()

This API is used to read the current pending interrupt status.

```
xspi_tgt_status_t xspi_tgt_int_status(xspi_tgt_dev_t *dev, uint32_t *out_status)
```

In/Out	Parameter	Description	Returns
In	*dev	Handle of the struct xspi_tgt_dev_t.	1: success 0: failure
Out	*out_status	Receives bitmask of pending interrupts.	

2.17. xspi_tgt_int_clear()

This API is used to clear pending interrupt bits (write-1-to-clear).

```
xspi_tgt_status_t xspi_tgt_int_clear(xspi_tgt_dev_t *dev, uint32_t mask)
```

In/Out	Parameter	Description	Returns
In	*dev	Handle of the struct xspi_tgt_dev_t.	1: success 0: failure
In	mask	OR-ed XSPI_TGT_INT_*_MSK bits to clear.	

2.18. xspi_tgt_simple_recv()

This API is used to handle a *Simple Protocol* SPI write transaction (polling).

```
xspi_tgt_status_t xspi_tgt_simple_recv(xspi_tgt_dev_t *dev, uint8_t *buf, size_t
buf_size, size_t  *out_len)
```

In/Out	Parameter	Description	Returns
In	*dev	Handle of the struct xspi_tgt_dev_t.	1: success 0: failure
Out	*buf	Destination buffer for received data.	
In	buf_size	Size of buffer in bytes.	
Out	*out_len	Receives the number of bytes read.	

2.19. xspi_tgt_simple_send()

This API is used to prepare Tx data for a *Simple Protocol* SPI read transaction.

```
xspi_tgt_status_t xspi_tgt_simple_send(xspi_tgt_dev_t *dev, const uint8_t *buf, size_t len)
```

In/Out	Parameter	Description	Returns
In	*dev	Handle of the struct xspi_tgt_dev_t.	1: success 0: failure
In	*buf	Source data buffer.	
In	len	Number of bytes to load into Tx FIFO.	

3. Function Call Flow Diagrams

3.1. xspi_tgt_init()

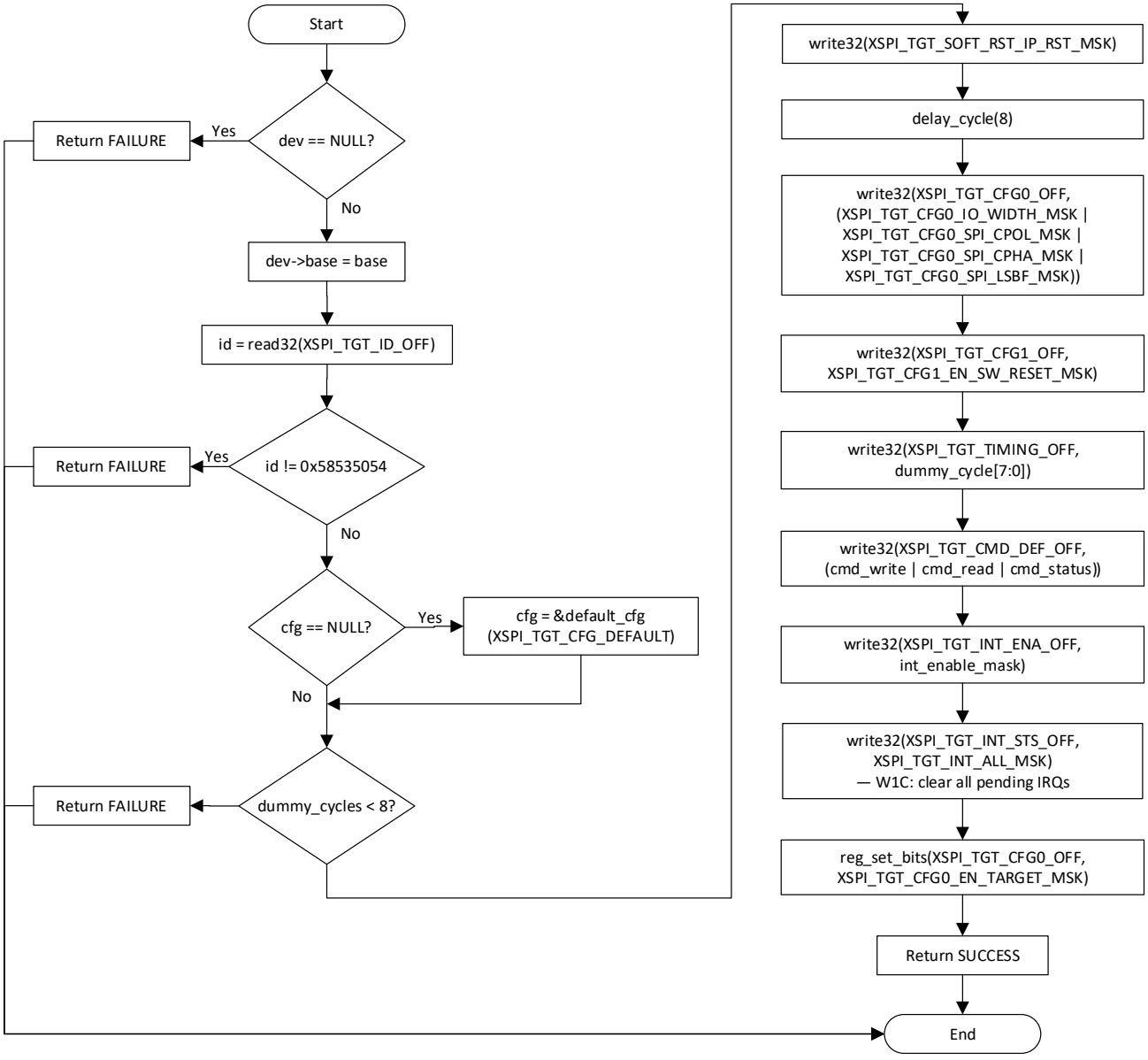

Figure 3.1. xspi_tgt_status_t xspi_tgt_init()

3.2. xspi_tgt_read_id()

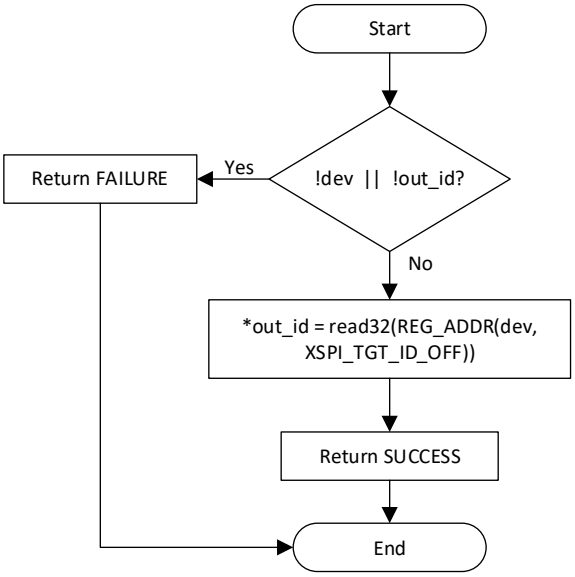

Figure 3.2. xspi_tgt_status_t xspi_tgt_read_id()

3.3. xspi_tgt_set_ready()

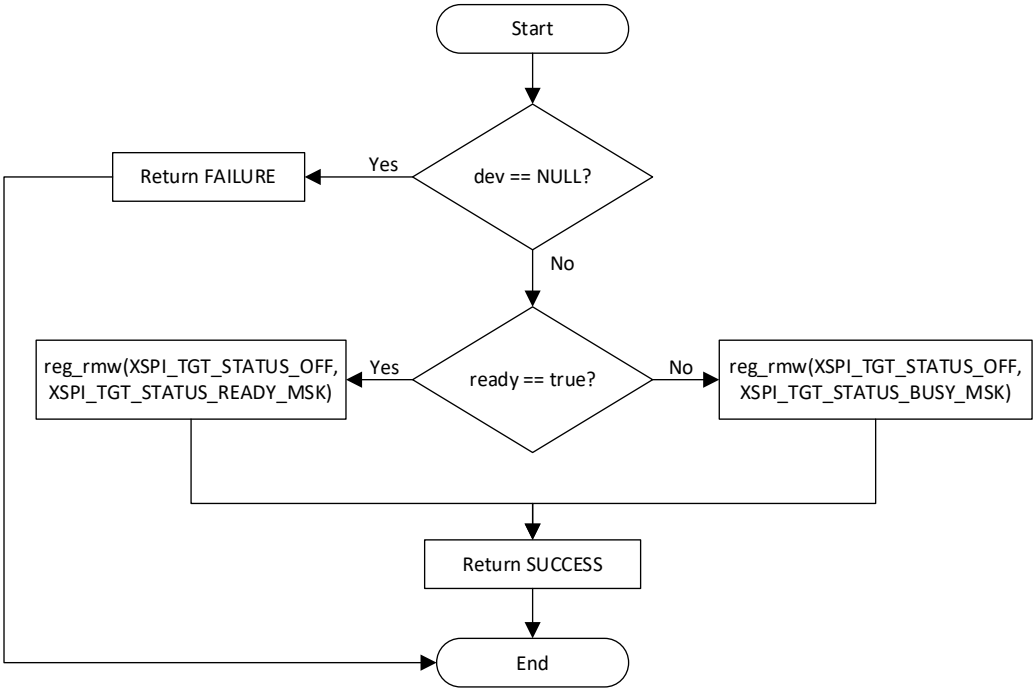

Figure 3.3. xspi_tgt_status_t xspi_tgt_set_ready()

3.4. xspi_tgt_get_first_byte()

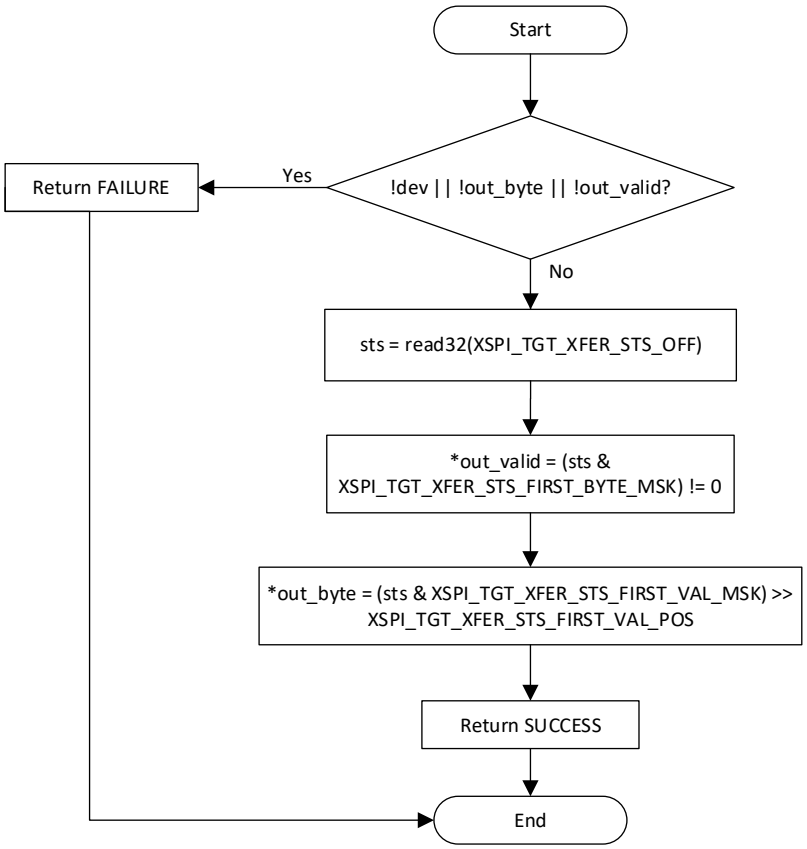

Figure 3.4. xspi_tgt_status_t xspi_tgt_get_first_byte()

3.5. xspi_tgt_tx_fifo_is_full()

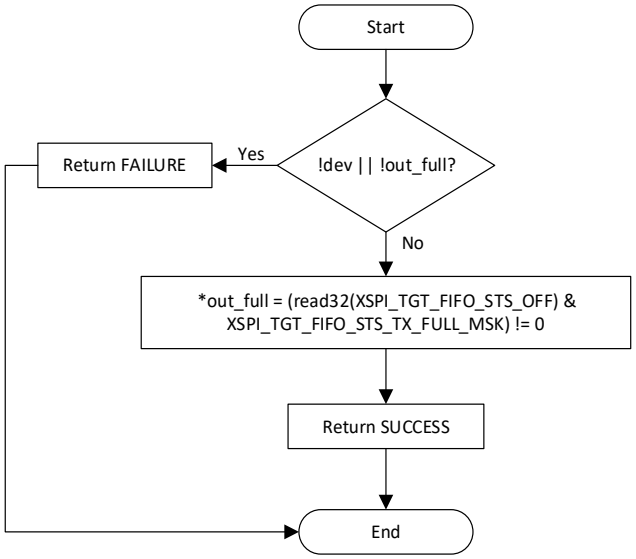

Figure 3.5. xspi_tgt_status_t xspi_tgt_tx_fifo_is_full()

3.6. xspi_tgt_tx_fifo_is_empty()

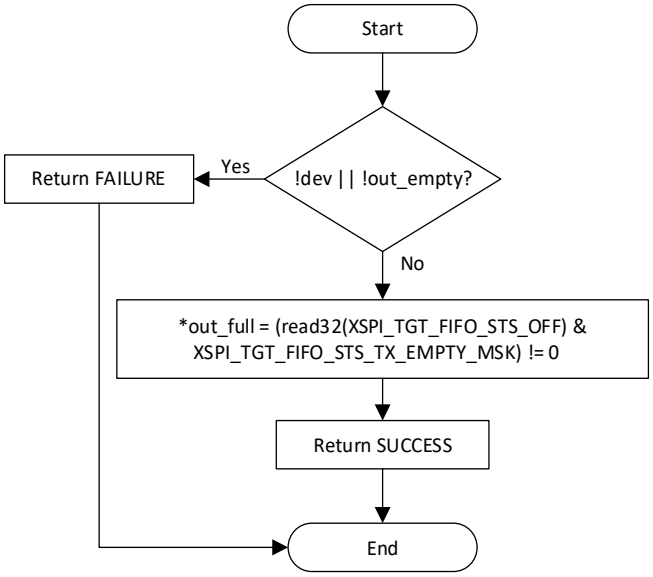

Figure 3.6. xspi_tgt_status_t xspi_tgt_tx_fifo_is_empty()

3.7. xspi_tgt_rx_fifo_is_full()

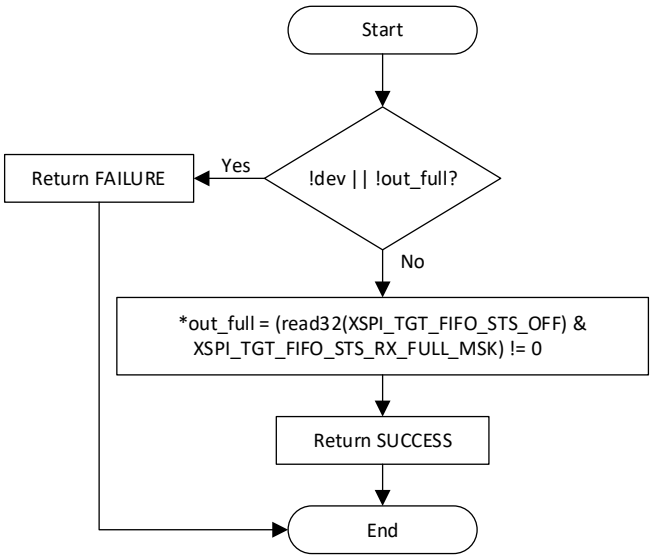

Figure 3.7. xspi_tgt_status_t xspi_tgt_rx_fifo_is_full()

3.8. xspi_tgt_rx_fifo_is_empty()

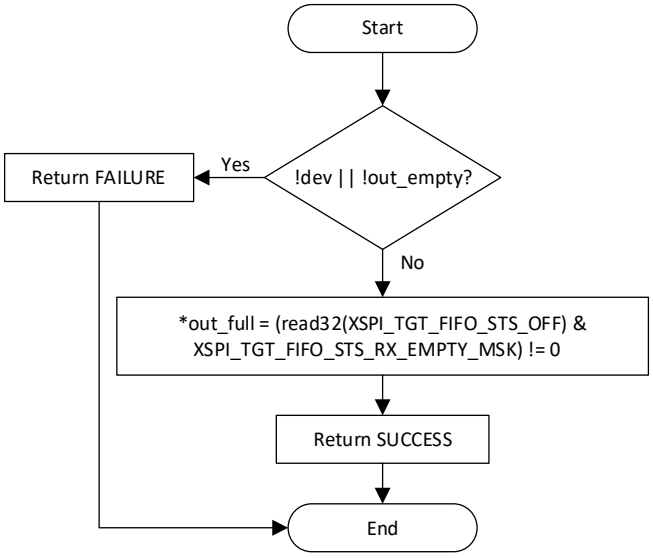

Figure 3.8. xspi_tgt_status_t xspi_tgt_rx_fifo_is_empty()

3.9. xspi_tgt_tx_byte()

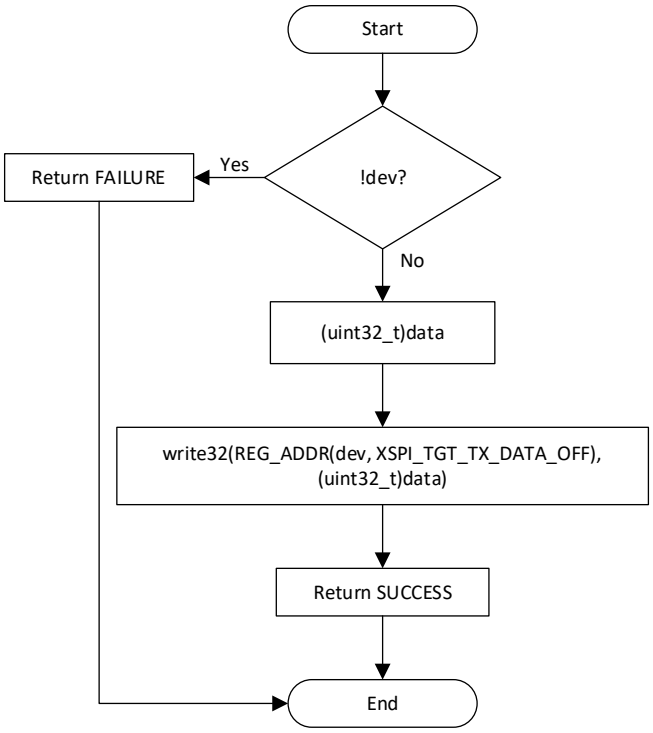

Figure 3.9. xspi_tgt_status_t xspi_tgt_tx_byte()

3.10. xspi_tgt_rx_byte()

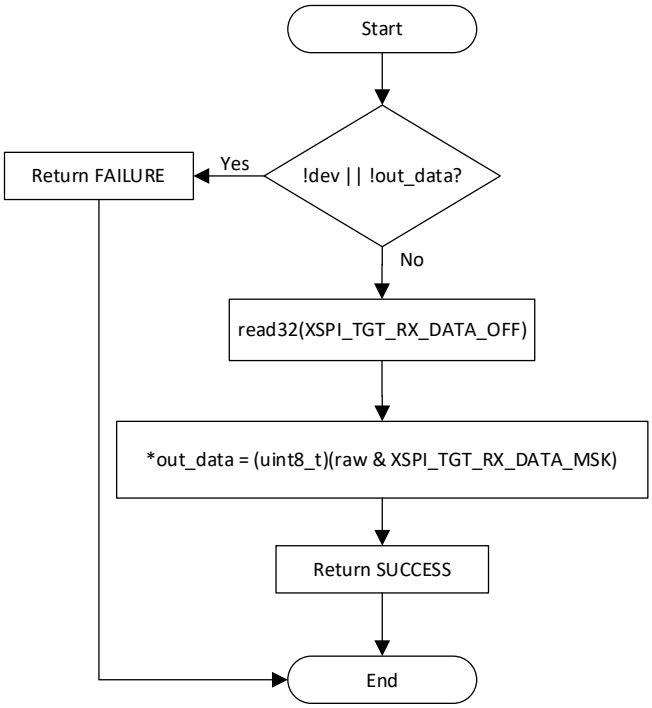

Figure 3.10. xspi_tgt_status_t xspi_tgt_rx_byte()

3.11. xspi_tgt_tx_write()

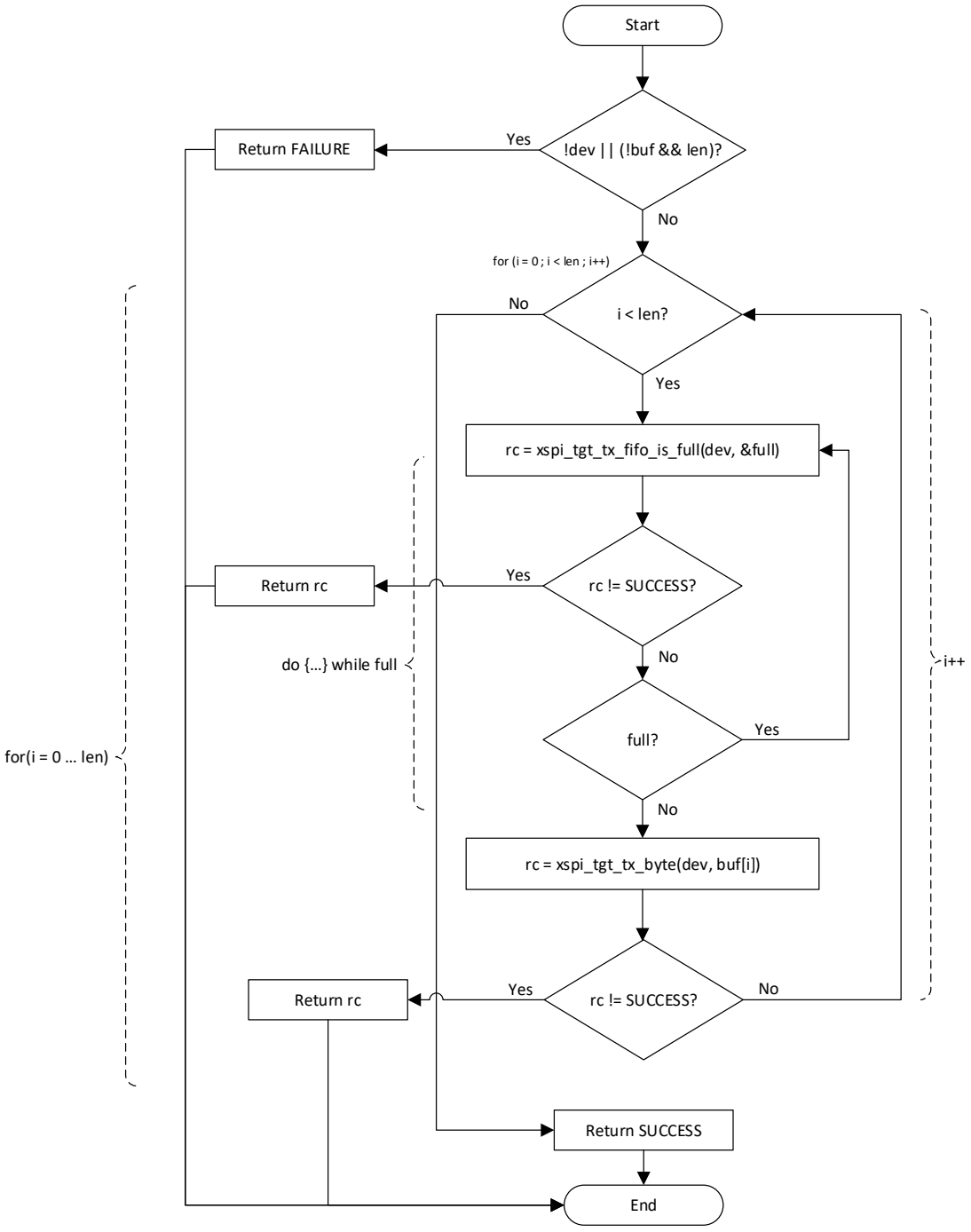

Figure 3.11. xspi_tgt_status_t xspi_tgt_tx_write()

3.12. xspi_tgt_soft_reset_ip()

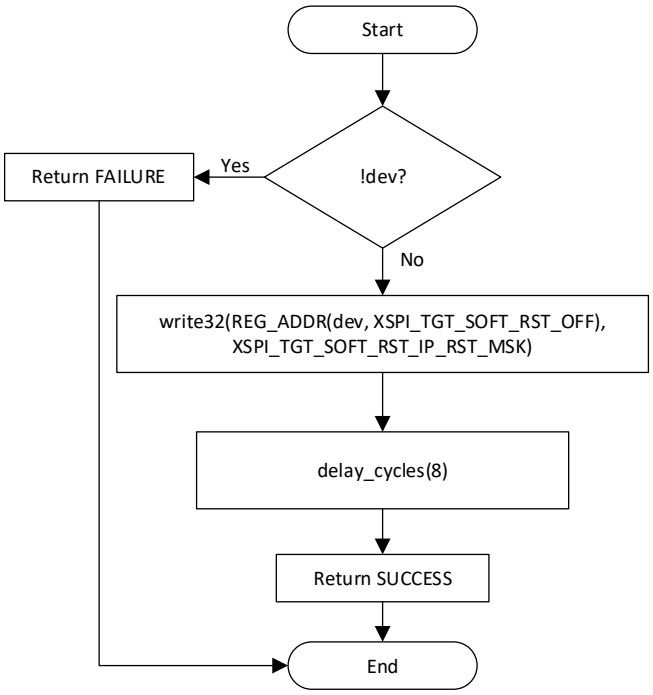

Figure 3.12. xspi_tgt_status_t xspi_tgt_soft_reset_ip()

3.13. xspi_tgt_soft_reset_tx_fifo()

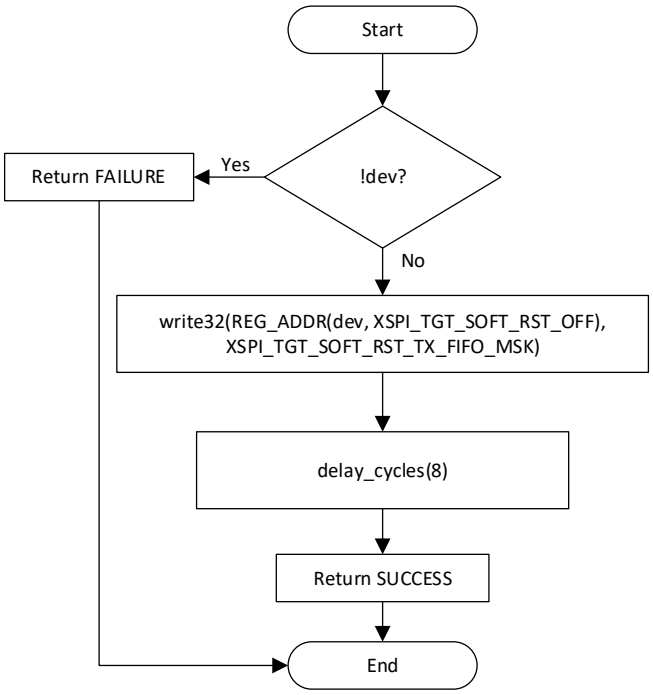

Figure 3.13. xspi_tgt_status_t xspi_tgt_soft_reset_tx_fifo()

3.14. xspi_tgt_soft_reset_rx_fifo()

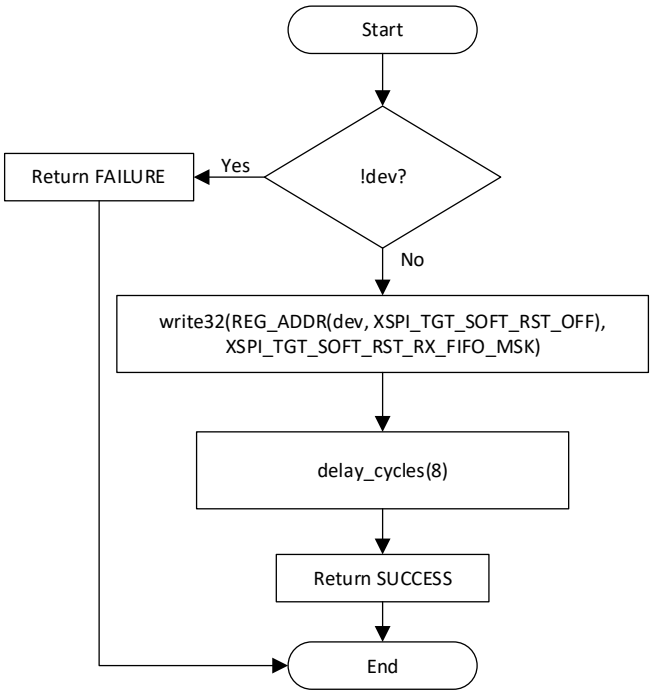

Figure 3.14. xspi_tgt_status_t xspi_tgt_soft_reset_rx_fifo()

3.15. xspi_tgt_int_enable()

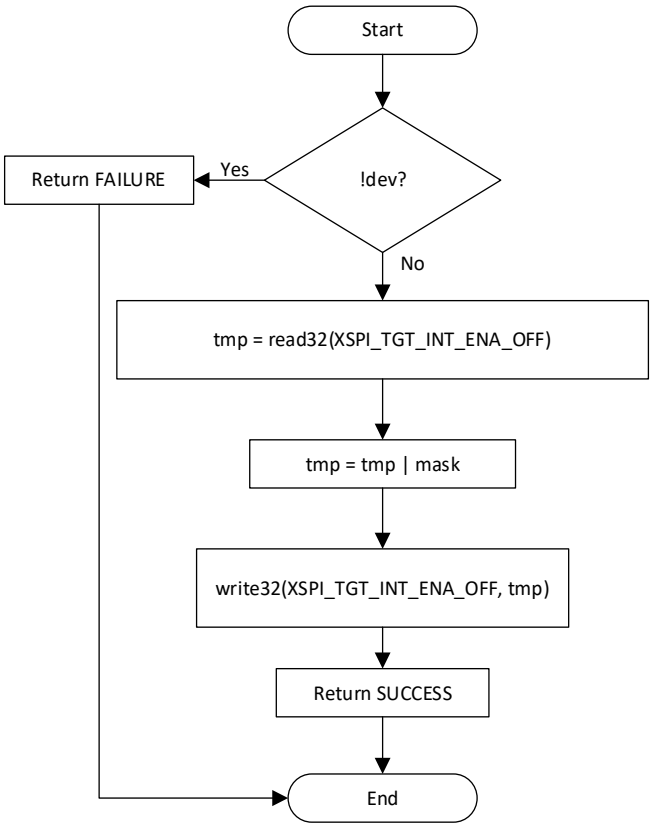

Figure 3.15. xspi_tgt_status_t xspi_tgt_int_enable()

3.16. xspi_tgt_int_status()

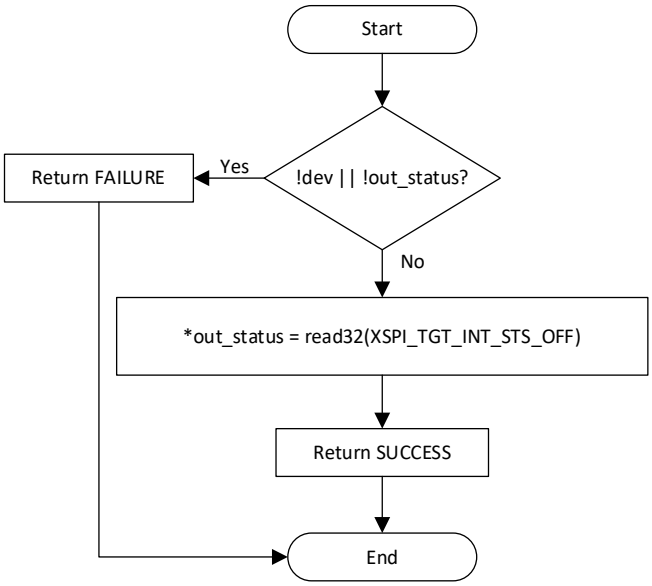

Figure 3.16. xspi_tgt_status_t xspi_tgt_int_status()

3.17. xspi_tgt_int_clear()

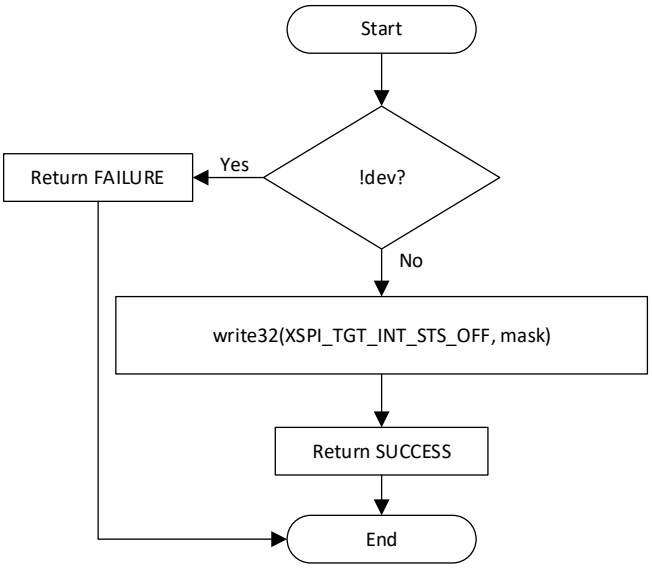

Figure 3.17. xspi_tgt_status_t xspi_tgt_int_clear()

3.18. xspi_tgt_simple_recv()

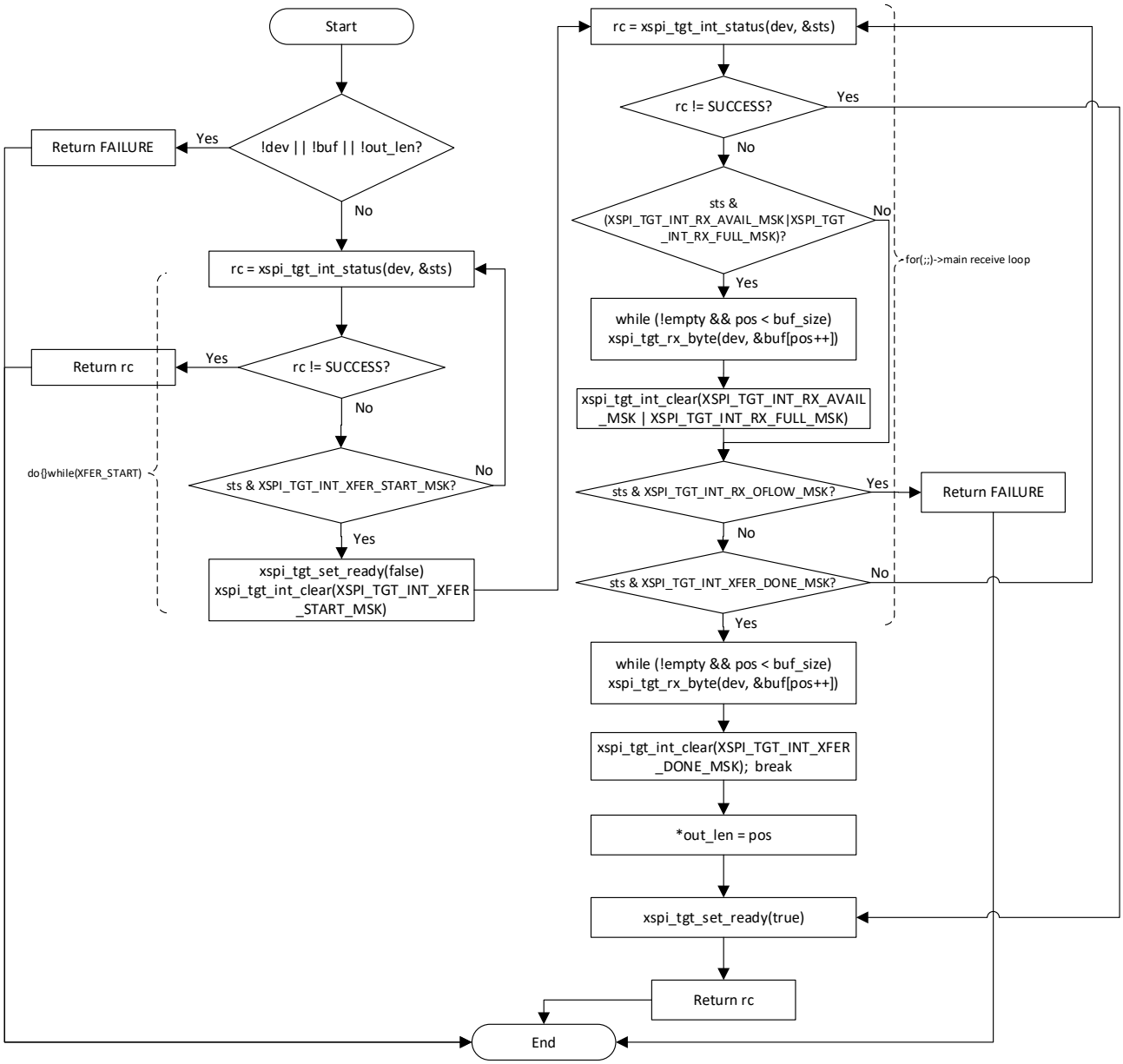

Figure 3.18. xspi_tgt_status_t xspi_tgt_simple_recv()

3.19. xspi_tgt_simple_send()

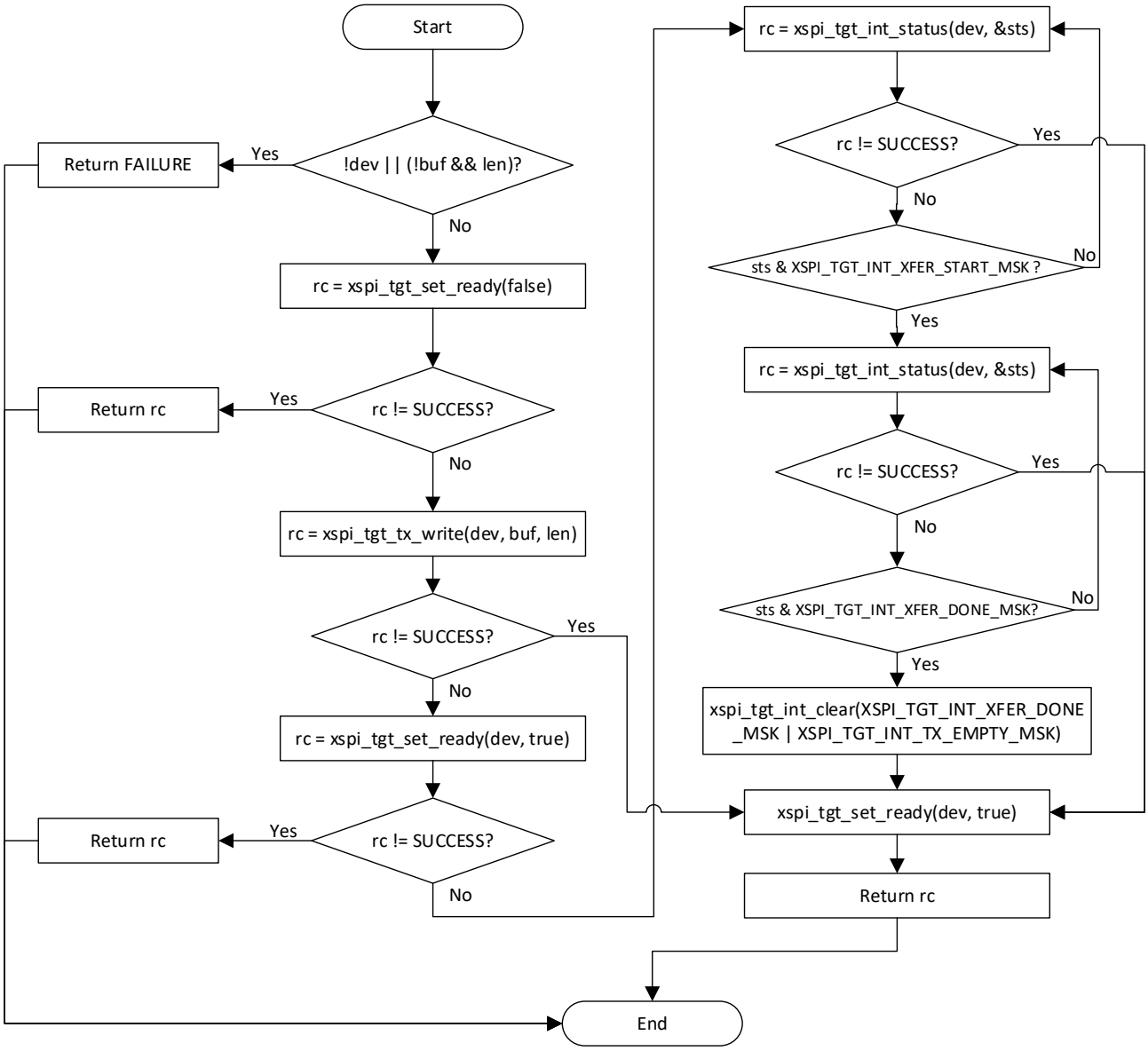

Figure 3.19. xspi_tgt_status_t xspi_tgt_simple_send()

4. API Data Structures

4.1. struct xspi_tgt_dev_t

Table 4.1. xspi_tgt_dev_t Parameters

Data Type	Struct Member	Description
unsigned int	base	Memory-mapped base address

4.2. struct xspi_tgt_cfg_t

Table 4.2. xspi_tgt_cfg_t Parameters

Data Type	Struct Member	Description
xspi_tgt_spi_mode_t	spi_mode	SPI mode 0 – SPI mode 0 (CPOL=0, CPHA=0) 1 – SPI mode 1 (CPOL=0, CPHA=1) 2 – SPI mode 2 (CPOL=1, CPHA=0) 3 – SPI mode 3 (CPOL=1, CPHA=1)
xspi_tgt_io_width_t	io_width	I/O width 0 – Standard/Single 1 – Quad
xspi_tgt_bit_order_t	bit_order	Bit order 0 – Most significant bit (MSB) 1 – Least significant bit (LSB)
uint8_t	dummy_cycles	Number of dummy cycles
bool	en_sw_reset	Enable in-band FFh reset
uint8_t	cmd_write	Preset command for write
uint8_t	cmd_read	Preset command for read
uint8_t	cmd_status	Preset command for status
uint32_t	int_enable_mask	Interrupt enable mask[1]

Note:

1. For information on the bits of the Interrupt Enable register (offset 0x010), refer to the xSPI Target IP User Guide (FPGA-IPUG-02323).

5. API Enum

5.1. enum xspi_tgt_spi_mode_t

Table 5.1. xspi_tgt_spi_mode_t Variables

Enum Member	Enum Decimal Value
XSPI_TGT_SPI_MODE0	0
XSPI_TGT_SPI_MODE1	1
XSPI_TGT_SPI_MODE2	2
XSPI_TGT_SPI_MODE3	3

5.2. enum xspi_tgt_io_width_t

Table 5.2. xspi_tgt_io_width_t Variables

Enum Member	Enum Decimal Value
XSPI_TGT_IO_X1	0
XSPI_TGT_IO_X4	2

5.3. enum xspi_tgt_bit_order_t

Table 5.3. xspi_tgt_bit_order_t Variables

Enum Member	Enum Decimal Value
XSPI_TGT_BIT_MSB	0
XSPI_TGT_BIT_LSB	1

6. API Macros

Table 6.1. API Macros Description

Macro	Description
#define SUCCESS	1
#define FAILURE	0
#define XSPI_TGT_ID_OFF	IP Identification register offset (0x000)
#define XSPI_TGT_CFG0_OFF	Configuration Register 0 offset (0x004)
#define XSPI_TGT_CFG1_OFF	Configuration Register 1 offset (0x008)
#define XSPI_TGT_TIMING_OFF	Timing Configuration register offset (0x00C)
#define XSPI_TGT_INT_ENA_OFF	Interrupt Enable register offset (0x010)
#define XSPI_TGT_CMD_DEF_OFF	Command Definition register offset (0x030)
#define XSPI_TGT_INT_STS_OFF	Interrupt Status register offset (0x100)
#define XSPI_TGT_XFER_STS_OFF	Transaction Status register offset (0x104)
#define XSPI_TGT_FIFO_STS_OFF	FIFO Status register offset (0x108)
#define XSPI_TGT_TX_DATA_OFF	Tx Data register offset (0x200)
#define XSPI_TGT_RX_DATA_OFF	Rx Data register offset (0x204)
#define XSPI_TGT_INT_SET_OFF	Interrupt Set (Debug) register offset (0x218)
#define XSPI_TGT_SOFT_RST_OFF	Soft Reset register offset (0x21C)
#define XSPI_TGT_STATUS_OFF	Target Status Auto Response register offset (0x22C)
#define XSPI_TGT_ID_VALUE	Expected identification (ID) value or IP unique identifier (0x58535054, *XSPT* in ASCII)
#define XSPI_TGT_CFG0_EN_TARGET_POS	Bit position for EN_TARGET (bit 0)
#define XSPI_TGT_CFG0_EN_TARGET_MSK	Bit mask for EN_TARGET
#define XSPI_TGT_CFG0_IO_WIDTH_POS	Bit position for IO_WIDTH (bit 4)
#define XSPI_TGT_CFG0_IO_WIDTH_MSK	Bit mask for IO_WIDTH (2 bits)
#define XSPI_TGT_CFG0_IO_WIDTH_X1	I/O width = x1 (single)
#define XSPI_TGT_CFG0_IO_WIDTH_X4	I/O width = x4 (quad)
#define XSPI_TGT_CFG0_SPI_CPOL_POS	Bit position for SPI_CPOL (bit 6)
#define XSPI_TGT_CFG0_SPI_CPOL_MSK	Bit mask for SPI_CPOL
#define XSPI_TGT_CFG0_SPI_CPHA_POS	Bit position for SPI_CPHA (bit 7)
#define XSPI_TGT_CFG0_SPI_CPHA_MSK	Bit mask for SPI_CPHA
#define XSPI_TGT_CFG0_SPI_LSBF_POS	Bit position for SPI_LSBF (bit 8)
#define XSPI_TGT_CFG0_SPI_LSBF_MSK	Bit mask for SPI_LSBF
#define XSPI_TGT_CFG1_EN_SW_RESET_POS	Bit position for EN_SW_RESET (bit 24)
#define XSPI_TGT_CFG1_EN_SW_RESET_MSK	Bit mask for EN_SW_RESET
#define XSPI_TGT_TIMING_DUMMY_CYCLES_POS	Bit position for DUMMY_CYCLES (bit 0)
#define XSPI_TGT_TIMING_DUMMY_CYCLES_MSK	Bit mask for DUMMY_CYCLES (8 bits)
#define XSPI_TGT_INT_XFER_DONE_POS	Bit position for XFER_DONE_INT (bit 0)
#define XSPI_TGT_INT_XFER_START_POS	Bit position for XFER_START_INT (bit 1)
#define XSPI_TGT_INT_TX_EMPTY_POS	Bit position for TX_EMPTY_INT (bit 2)
#define XSPI_TGT_INT_TX_UFLOW_POS	Bit position for TX_UFLOW_INT (bit 3)
#define XSPI_TGT_INT_RX_AVAIL_POS	Bit position for RX_AVAIL_INT (bit 4)
#define XSPI_TGT_INT_RX_FULL_POS	Bit position for RX_FULL_INT (bit 5)
#define XSPI_TGT_INT_RX_OFLOW_POS	Bit position for RX_OFLOW_INT (bit 6)
#define XSPI_TGT_INT_ERR_POS	Bit position for ERR_INT (bit 7)
#define XSPI_TGT_INT_RESET_DET_POS	Bit position for RESET_DET_INT (bit 10)
#define XSPI_TGT_INT_WR_ON_TX_FULL_POS	Bit position for WR_ON_TX_FULL_INT (bit 16)
#define XSPI_TGT_INT_RD_ON_RX_EMPTY_POS	Bit position for RD_ON_RX_EMPTY_INT (bit 17)

Macro	Description
#define XSPI_TGT_INT_XFER_DONE_MSK	Bit mask for XFER_DONE_INT
#define XSPI_TGT_INT_XFER_START_MSK	Bit mask for XFER_START_INT
#define XSPI_TGT_INT_TX_EMPTY_MSK	Bit mask for TX_EMPTY_INT
#define XSPI_TGT_INT_TX_UFLOW_MSK	Bit mask for TX_UFLOW_INT
#define XSPI_TGT_INT_RX_AVAIL_MSK	Bit mask for RX_AVAIL_INT
#define XSPI_TGT_INT_RX_FULL_MSK	Bit mask for RX_FULL_INT
#define XSPI_TGT_INT_RX_OFLOW_MSK	Bit mask for RX_OFLOW_INT
#define XSPI_TGT_INT_ERR_MSK	Bit mask for ERR_INT
#define XSPI_TGT_INT_RESET_DET_MSK	Bit mask for RESET_DET_INT
#define XSPI_TGT_INT_WR_ON_TX_FULL_MSK	Bit mask for WR_ON_TX_FULL_INT
#define XSPI_TGT_INT_RD_ON_RX_EMPTY_MSK	Bit mask for RD_ON_RX_EMPTY_INT
#define XSPI_TGT_INT_ALL_MSK	Combined mask of all interrupt bits
#define XSPI_TGT_CMD_DEF_WRITE_POS	Bit position for write command opcode (bit 0)
#define XSPI_TGT_CMD_DEF_WRITE_MSK	Bit mask for write command opcode (8 bits)
#define XSPI_TGT_CMD_DEF_READ_POS	Bit position for read command opcode (bit 8)
#define XSPI_TGT_CMD_DEF_READ_MSK	Bit mask for read command opcode (8 bits)
#define XSPI_TGT_CMD_DEF_STATUS_POS	Bit position for status read command opcode (bit 16)
#define XSPI_TGT_CMD_DEF_STATUS_MSK	Bit mask for status command opcode (8 bits)
#define XSPI_TGT_XFER_STS_ACTIVE_POS	Bit position for XFER_ACTIVE (bit 0)
#define XSPI_TGT_XFER_STS_ACTIVE_MSK	Bit mask for XFER_ACTIVE
#define XSPI_TGT_XFER_STS_FIRST_BYTE_POS	Bit position for FIRST_BYTE (bit 1)
#define XSPI_TGT_XFER_STS_FIRST_BYTE_MSK	Bit mask for FIRST_BYTE
#define XSPI_TGT_XFER_STS_FIRST_VAL_POS	Bit position for FIRST_BYTE_VAL (bit 8)
#define XSPI_TGT_XFER_STS_FIRST_VAL_MSK	Bit mask for FIRST_BYTE_VAL (8 bits)
#define XSPI_TGT_XFER_STS_IO_MODE_POS	Bit position for IO_MODE (bit 16)
#define XSPI_TGT_XFER_STS_IO_MODE_MSK	Bit mask for IO_MODE (2 bits)
#define XSPI_TGT_XFER_STS_ERR_CODE_POS	Bit position for ERR_CODE (bit 24)
#define XSPI_TGT_XFER_STS_ERR_CODE_MSK	Bit mask for ERR_CODE (4 bits)
#define XSPI_TGT_FIFO_STS_TX_EMPTY_POS	Bit position for TX_FIFO_EMPTY (bit 10)
#define XSPI_TGT_FIFO_STS_TX_EMPTY_MSK	Bit mask for TX_FIFO_EMPTY
#define XSPI_TGT_FIFO_STS_TX_FULL_POS	Bit position for TX_FIFO_FULL (bit 11)
#define XSPI_TGT_FIFO_STS_TX_FULL_MSK	Bit mask for TX_FIFO_FULL
#define XSPI_TGT_FIFO_STS_RX_EMPTY_POS	Bit position for RX_FIFO_EMPTY (bit 26)
#define XSPI_TGT_FIFO_STS_RX_EMPTY_MSK	Bit mask for RX_FIFO_EMPTY
#define XSPI_TGT_FIFO_STS_RX_FULL_POS	Bit position for RX_FIFO_FULL (bit 27)
#define XSPI_TGT_FIFO_STS_RX_FULL_MSK	Bit mask for RX_FIFO_FULL
#define XSPI_TGT_TX_DATA_MSK	Data mask for Tx data register (8 bits, 0xFF)
#define XSPI_TGT_RX_DATA_MSK	Data mask for Rx data register (8 bits, 0xFF)
#define XSPI_TGT_SOFT_RST_IP_RST_POS	Bit position for IP RST (bit 0)
#define XSPI_TGT_SOFT_RST_IP_RST_MSK	Bit mask for IP_RST
#define XSPI_TGT_SOFT_RST_TX_FIFO_POS	Bit position for TX_FIFO_RST (bit 1)
#define XSPI_TGT_SOFT_RST_TX_FIFO_MSK	Bit mask for TX_FIFO_RST
#define XSPI_TGT_SOFT_RST_RX_FIFO_POS	Bit position for RX_FIFO_RST (bit 2)
#define XSPI_TGT_SOFT_RST_RX_FIFO_MSK	Bit mask for RX_FIFO_RST
#define XSPI_TGT_SOFT_RST_CSR_POS	Bit position for CSR_RST (bit 3)
#define XSPI_TGT_SOFT_RST_CSR_MSK	Bit mask for CSR_RST
#define XSPI_TGT_STATUS_BUSY_POS	Bit position for busy flag (bit 0)
#define XSPI_TGT_STATUS_BUSY_MSK	Bit mask for busy flag

Macro	Description
#define XSPI_TGT_STATUS_ACK_POS	Bit position for acknowledge flag (bit 1)
#define XSPI_TGT_STATUS_ACK_MSK	Bit mask for acknowledge flag
#define XSPI_TGT_STATUS_ERR_POS	Bit position for error flag (bit 2)
#define XSPI_TGT_STATUS_ERR_MSK	Bit mask for error flag
#define XSPI_TGT_STATUS_ERR_CODE_POS	Bit position for error code (bit 3)
#define XSPI_TGT_STATUS_ERR_CODE_MSK	Bit mask for error code (3 bits)
#define XSPI_TGT_STATUS_READY_POS	Bit position for ready flag (bit 7)
#define XSPI_TGT_STATUS_READY_MSK	Bit mask for ready flag
#define XSPI_TGT_REG(base, off)	Build a volatile 32-bit register pointer from base address + offset
#define XSPI_TGT_CFG_DEFAULT	Default configuration for the struct xspi_tgt_cfg_t: spi_mode = XSPI_TGT_SPI_MODE0 io_width = XSPI_TGT_IO_X1 bit_order = XSPI_TGT_BIT_MSB dummy_cycles = 8 en_sw_reset = false cmd_write = 0x02 cmd_read = 0x03 cmd_status = 0x05 int_enable_mask = 0

References

- xSPI Target IP User Guide (FPGA-IPUG-02323)
- xSPI Target IP Release Notes (FPGA-RN-02115)
- Certus-NX web page
- CertusPro-NX web page
- CrossLink-NX web page
- MachXO5-NX web page
- MachXO3D 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 1.0, June 2026

Section	Change Summary
All	Initial release.

www.latticesemi.com