Octal SPI Controller Driver API Reference

Technical Note

FPGA-TN-02400-1.2

December 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
AXI4	Advanced eXtensible Interface 4
CSR	Configuration Status Register
DDR	Double Data Rate
DSPI	Dual Serial Peripheral Interface
DTR	Double Transfer Rate
DWORD	Double Word
FIFO	First In First Out
FPGA	Field Programmable Gate Array
ID	Identification
I/O	Input/Output
IP	Intellectual Property
LSB	Least Significant Bit
QPI	Quad Peripheral Interface (4-4-4 Protocol)
QSPI	Quad Serial Peripheral Interface
Rx	Receiver
SPI	Serial Peripheral Interface
STR	Single Transfer Rate
Tx	Transmitter
XiP	eXecute In Place
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 Lattice Octal SPI Controller IP is an SPI interface that supports different types of SPI protocols: standard, dual, quad, and xSPI. Standard SPI is the legacy four-wire interface with separate data lines for input and output. Dual SPI (DSPI) is an extension of standard SPI using two data lines. Quad SPI (QSPI) is another extension of standard SPI designed to provide a higher data transfer rate using four data lines. Expanded SPI (xSPI) is an advanced extension of standard SPI designed to provide higher data throughput and improved performance while maintaining backward compatibility with standard SPI devices. xSPI uses eight data lines and is capable of double data rate transfer. This document describes the APIs used with the octal SPI controller driver.

Refer to the Octal SPI Controller IP User Guide (FPGA-IPUG-02273) 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 the Lattice CertusPro™-NX, Certus™-NX, and Avant™ (LAV-AT-E70, LAV-AT-G70, LAV-AT-X70) devices. The technical guide assumes readers have expertise in embedded systems and FPGA technologies.

1.3. Driver Version

OCTAL_SPI_CONTROLLER_DRV_VER “25.2.0”

1.4. Driver and IP Compatibility

Driver Version	IP Version
25.2.0	1.4.0

Refer to the Octal SPI Controller IP Release Notes (FPGA-RN-02011) for more information on the driver and IP versions.

1.5. Flash Device Selection

Before using the Octal SPI Controller IP, you must define the flash device to be used. The Octal SPI Controller driver supports three SPI flash devices:

- Winbond™ W25Q512JV
- Macronix™ MX25L51245G
- Micron™ MT25QU128ABA

By default, the Micron SPI flash device is selected.

To select a different SPI flash device:

1. Open the octal_spi_controller.h header file located at <project directory>/src/bsp/driver/octal_spi_controller/.
2. Locate the section labeled *Select SPI device.*
3. Uncomment the line corresponding to the desired SPI flash device by removing the // at the beginning of the line.
4. Comment out the currently selected device by adding // at the beginning of its line.

Note: Only one SPI flash device must be selected at a time.

The following code snippet shows the *Select SPI device* section in the octal_spi_controller.h header file:

```
// ------------------------------------------------
// Select SPI device
#define _MICRON_X4_FLASH_DEVICE_
//#define _WINBOND_X4_FLASH_DEVICE_
//#define _MACRONIX_X4_FLASH_DEVICE_
// ------------------------------------------------
```

2. API Description

2.1. spix8_flash_ctl_init()

This API initializes the Octal SPI Controller IP with the following actions:

- Assigns the base address to the spix8_ctl_handle_t struct.
- Initializes the Octal IP configuration derived from the input pointer.

```
unsigned char spix8_flash_ctl_init(spix8_ctl_handle_t *handle, unsigned int base_addr,
unsigned int flash_addr_offset)
```

In/Out	Parameter	Description	Returns
In	handle	Handle of the struct spix8_ctl_handle_t	0: Success 1: Failure
In	base_addr	Base address to be assigned	
In	flash_addr_offset	Value for target start address offset register (offset 0x30)	

2.2. set_spi_clock_freq()

This API is used to set the SPI clock frequency.

```
unsigned char set_spi_clock_freq(spix8_ctl_handle_t *handle, unsigned int sck_freq)
```

In/Out	Parameter	Description	Returns
In	handle	Handle of the struct spix8_ctl_handle_t	0: Success 1: Failure
In	sck_freq	Clock frequency to set (MHz)	

2.3. gencmd_flash_rdfsr()

This API is used to read the register flag status (for Micron SPI flash only).

```
unsigned char gencmd_flash_rdfsr(spix8_ctl_handle_t *handle, unsigned int *rdat)
```

In/Out	Parameter	Description	Returns
In	handle	Handle of the struct spix8_ctl_handle_t	0: Success 1: Failure
In	rdat	Pointer to store the flag status value	

2.4. gencmd_flash_rdid()

This API is used to read the SPI flash device ID.

```
unsigned char gencmd_flash_rdid(spix8_ctl_handle_t *handle, unsigned int *rdat)
```

In/Out	Parameter	Description	Returns
In	handle	Handle of the struct spix8_ctl_handle_t	0: Success 1: Failure
In	rdat	Pointer to store the ID value	

2.5. gencmd_flash_wrsr123()

This API is used to write data into Winbond register 1, 2, or 3 (hexadecimal in datasheet of 0x1, 0x31, or 0x11).

```
unsigned char gencmd_flash_wrsr123(spix8_ctl_handle_t *handle, unsigned int reg_data,
unsigned int sr_type, unsigned int reg_type)
```

In/Out	Parameter	Description	Returns
In	handle	Handle of the struct spix8_ctl_handle_t	0: Success 1: Failure
In	reg_data	Register data to be written	
In	sr_type	Register type 0 – FLASH_CMD_WRSR (register 1) 1 – FLASH_CMD_WRSR2 (register 2) 2 – FLASH_CMD_WRSR3 (register 3)	
In	reg_type	Write register type 0 – FLASH_REG_VOLATILE 1 – FLASH_REG_NON_VOLATILE 2 – FLASH_REG_STATUS 3 – FLASH_REG_PROT_MGMT 4 – FLASH_REG_GEN_PURPOSE	

2.6. gencmd_flash_rdsr123()

This API is used to read data from Winbond register 1, 2, or 3 (hexadecimal in datasheet of 0x1, 0x31, or 0x11).

```
unsigned char gencmd_flash_rdsr123(spix8_ctl_handle_t *handle, unsigned int *rdat,
unsigned int sr_type)
```

In/Out	Parameter	Description	Returns
In	handle	Handle of the struct spix8_ctl_handle_t	0: Success 1: Failure
In	rdat	Pointer to store the register data read from SPI flash	
In	sr_type	Register type 0 – FLASH_CMD_WRSR (register 1) 1 – FLASH_CMD_WRSR2 (register 2) 2 – FLASH_CMD_WRSR3 (register 3)	

2.7. gencmd_flash_set_4byte_mode()

This API is used to set the SPI flash address mode.

```
unsigned char gencmd_flash_set_4byte_mode(spix8_ctl_handle_t *handle, unsigned int
adr_mode)
```

In/Out	Parameter	Description	Returns
In	handle	Handle of the struct spix8_ctl_handle_t	0: Success 1: Failure
In	adr_mode	Set address mode 0 – Exit 4-byte mode 1 – Enter 4-byte mode	

2.8. gencmd_flash_set_quad_mode()

This API is used to enter quad mode.

```
unsigned char gencmd_flash_set_quad_mode(spix8_ctl_handle_t *handle, unsigned int
en_quad)
```

In/Out	Parameter	Description	Returns
In	handle	Handle of the struct spix8_ctl_handle_t	0: Success 1: Failure
In	en_quad	Enable or disable quad mode 0 – Disable quad mode (fall back into x1 mode) 1 – Enable quad mode	

2.9. gencmd_flash_soft_reset()

This API is used to perform a soft reset in the external SPI flash.

```
unsigned char gencmd_flash_soft_reset(spix8_ctl_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	handle	Handle of the struct spix8_ctl_handle_t	0: Success 1: Failure

2.10. gencmd_quadspi_4kb_erase()

This API is used to perform a 4-kb erase in QSPI mode.

```
unsigned char gencmd_quadspi_4kb_erase(spix8_ctl_handle_t *handle, unsigned int
start_addr)
```

In/Out	Parameter	Description	Returns
In	handle	Handle of the struct spix8_ctl_handle_t	0: Success 1: Failure
In	start_addr	Start address of the SPI flash to be erased	

2.11. gencmd_quadspi_page_program()

This API is used to perform a page program in SPI flash.

```
unsigned char gencmd_quadspi_page_program(spix8_ctl_handle_t *handle, unsigned int
num_bytes, unsigned int *dat_buf, unsigned int start_addr)
```

In/Out	Parameter	Description	Returns
In	handle	Handle of the struct spix8_ctl_handle_t	0: Success 1: Failure
In	num_bytes	Number of bytes to be programed	
In	dat_buffer	Data buffer pointer to be programed into SPI flash	
In	start_addr	Start address of the SPI flash to be programed	

2.12. config_supported_flash_cmd()

This API is used to configure the supported flash commands.

```
unsigned char config_supported_flash_cmd(spix8_ctl_handle_t *handle,
unsigned int prog_type, unsigned int read_type, unsigned int xip)
```

In/Out	Parameter	Description	Returns
In	handle	Handle of the struct spix8_ctl_handle_t	0: Success 1: Failure
In	prog_type	Program type 0 – FLASH_PAGE_PROG (page program) 1 – FLASH_OCTIN_FAST_PROG (fast program) 2 – FLASH_EXT_OCTIN_FAST_PROG (extended fast program)	
In	read_type	Read type 0 – FLASH_READ (standard flash read) 1 – FLASH_FAST_READ (fast read) 2 – FLASH_OCTOUT_FAST_READ (octal output fast read) 3 – FLASH_OCT_IO_FAST_READ (octal input/output fast read) 4 – FLASH_DDR_OCTOUT_FAST_READ (DDR octal output fast read) 5 – FLASH_DDR_OCT_IO_FAST_READ (DDR octal input/output fast read)	
In	xip	Option to enable XiP feature	

2.13. gencmd_flash_rdsr()

This API is used to read the configuration flag status register from SPI flash.

```
unsigned char gencmd_flash_rdsr(spix8_ctl_handle_t *handle, unsigned int *rdat)
```

In/Out	Parameter	Description	Returns
In	handle	Handle of the struct spix8_ctl_handle_t	0: Success 1: Failure
In	rdat	Data pointer to store register data read from SPI flash	

2.14. set_quad_flash_command_config()

This API is used to set the supported command configuration register (offset 0x18).

```
unsigned char set_quad_flash_command_config(spix8_ctl_handle_t *handle, unsigned int prog_type, unsigned int read_type, unsigned int wr1_rd0, unsigned char xip)
```

In/Out	Parameter	Description	Returns
In	handle	Handle of the struct spix8_ctl_handle_t	0: Success 1: Failure
In	prog_type	Program type 0 – FLASH_PAGE_PROG (page program) 1 – FLASH_QUAD_FAST_PROG (quad fast program) 2 – FLASH_EXT_QUAD_FAST_PROG (extended quad fast program)	
In	read_type	Read type 0 – FLASH_READ (standard flash read) 1 – FLASH_FAST_READ (fast read) 2 – FLASH_FAST_READ_2DO (dual output) 3 – FLASH_FAST_READ_2IO (dual input/output) 4 – FLASH_FAST_READ_4DO (quad output) 5 – FLASH_FAST_READ_4IO (quad input/output) 6 – FLASH_FAST_READ_1IODTR (single input/output double transfer rate) 7 – FLASH_FAST_READ_2IODTR (dual input/output double transfer rate) 8 – FLASH_FAST_READ_4IODTR (quad input/output double transfer rate)	
In	wr1_rd0	Write or read command 0 – Read only 1 – Write only	
In	xip	Option to enable XiP feature	

2.15. gencmd_quadspi_fast_read()

This API is used to perform the QSPI fast read action in SPI flash.

```
unsigned char gencmd_quadspi_fast_read(spix8_ctl_handle_t *handle, unsigned int num_bytes, unsigned int *dat_buf, unsigned int start_addr)
```

In/Out	Parameter	Description	Returns
In	handle	Handle of the struct spix8_ctl_handle_t	0: Success 1: Failure
In	num_bytes	Number of data (4-byte basis) to read	
In	dat_buf	Pointer to store the read back data	
In	start_addr	Start address of data to read	

2.16. quad_xip_activate()

This API is used to activate the XiP feature of the SPI flash.

```
unsigned char quad_xip_activate(spix8_ctl_handle_t *handle, unsigned int read_type)
```

In/Out	Parameter	Description	Returns
In	handle	Handle of the struct spix8_ctl_handle_t	0: Success 1: Failure
In	read_type	Read type 0 – FLASH_READ (standard flash read) 1 – FLASH_FAST_READ (fast read) 2 – FLASH_OCTOUT_FAST_READ (octal output fast read) 3 – FLASH_OCT_IO_FAST_READ (octal input/output fast read) 4 – FLASH_DDR_OCTOUT_FAST_READ (DDR octal output fast read) 5 – FLASH_DDR_OCT_IO_FAST_READ (DDR octal input/output fast read)	

2.17. gencmd_wr_vcr_set_octal_mode()

This API is used to configure flash registers to octal SPI mode and set the spix8_ctl_handle_t struct members to octal SPI configurations.

```
uint32_t gencmd_wr_vcr_set_octal_mode(spix8_ctl_handle_t *handle, uint32_t en_octal)
```

In/Out	Parameter	Description	Returns
In	handle	Handle of the struct spix8_ctl_handle_t	0: Success 1: Failure
In	en_octal	Enable or disable octal SPI 0 – Disable 1 – Enable	

2.18. gencmd_octspi_erase()

This API is used to perform an erase operation in octal SPI mode.

```
unsigned char gencmd_octspi_erase(spix8_ctl_handle_t *handle, unsigned char erase_type,
unsigned int start_addr)
```

In/Out	Parameter	Description	Returns
In	handle	Handle of the struct spix8_ctl_handle_t	0: Success 1: Failure
In	erase_type	Erase type to perform 0 – FLASH_ERASE_4KB (4 kb erase) 1 – FLASH_ERASE_32KB (32 kb erase) 2 – FLASH_ERASE_128KB (128 kb erase) 3 – FLASH_ERASE_CHIP (chip erase)	
In	start_addr	Start address to perform erase operation	

2.19. gencmd_octspi_fast_read()

This API is used to perform a fast read operation in octal SPI mode.

```
unsigned char gencmd_octspi_fast_read(spix8_ctl_handle_t *handle, unsigned int
num_bytes, unsigned int *dat_buf, unsigned int start_addr)
```

In/Out	Parameter	Description	Returns
In	handle	Handle of the struct spix8_ctl_handle_t	0: Success 1: Failure
In	num_bytes	Number of data (4-byte basis) to be read	
In	dat_buf	Pointer to store the read back data	
In	start_addr	Start address to perform fast read operation	

2.20. gencmd_octspi_page_program()

This API is used to perform a page program in octal SPI mode.

```
unsigned char gencmd_octspi_page_program(spix8_ctl_handle_t *handle, unsigned int
num_bytes, unsigned int *dat_buf, unsigned int start_addr)
```

In/Out	Parameter	Description	Returns
In	handle	Handle of the struct spix8_ctl_handle_t	0: Success 1: Failure
In	num_bytes	Number of data (4-byte basis) to be programmed	
In	dat_buf	Pointer to store the data to be programmed	
In	start_addr	Start address to perform program operation	

3. Function Call Flow Diagrams

3.1. spix8_flash_ctl_init()

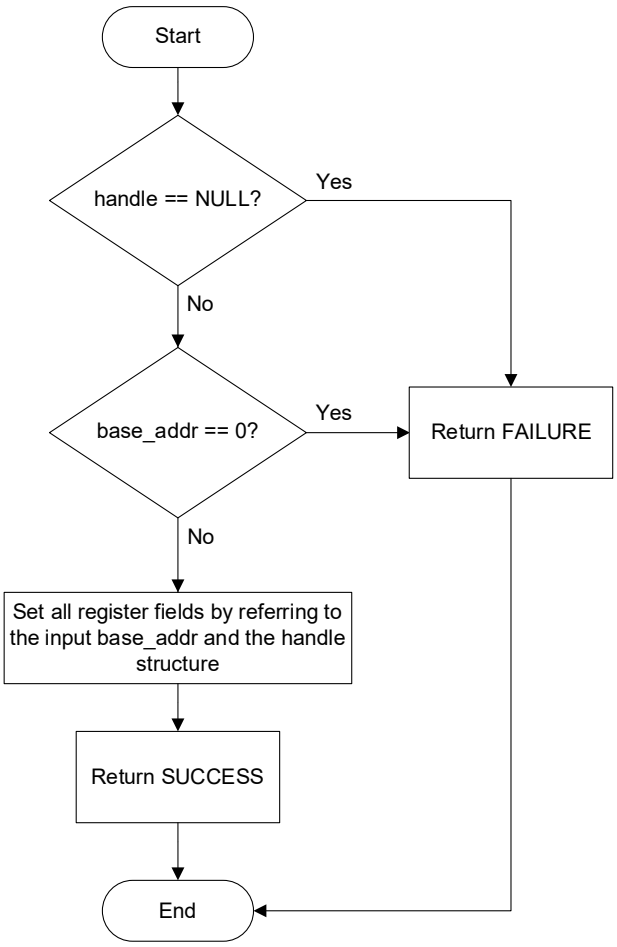

Figure 3.1. unsigned char spix8_flash_ctl_init()

3.2. set_spi_clock_freq()

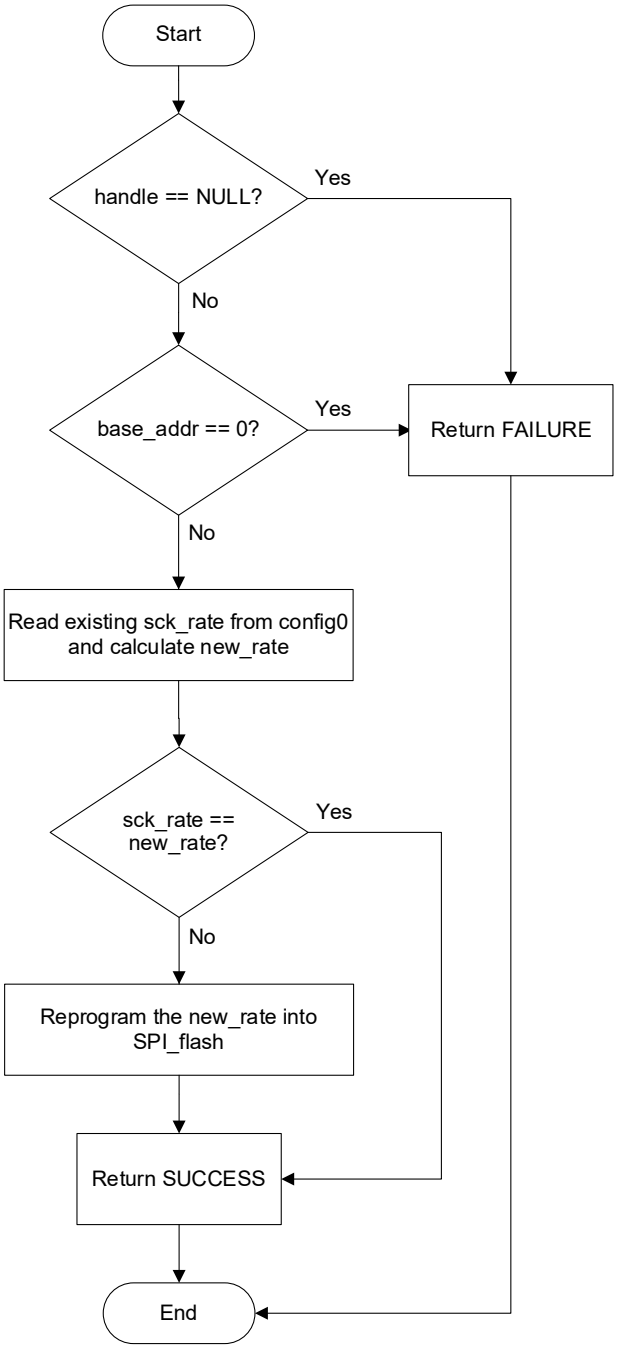

Figure 3.2. unsigned char set_spi_clock_freq()

3.3. gencmd_flash_rdfsr()

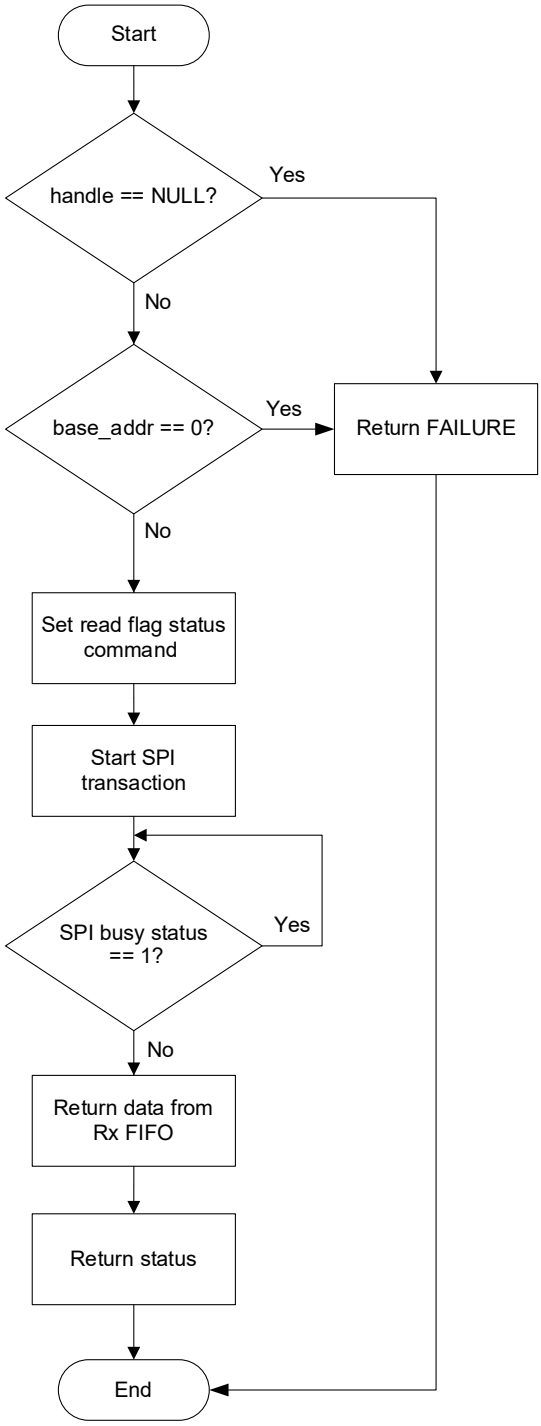

Figure 3.3. unsigned char gencmd_flash_rdfsr()

3.4. gencmd_flash_rdid()

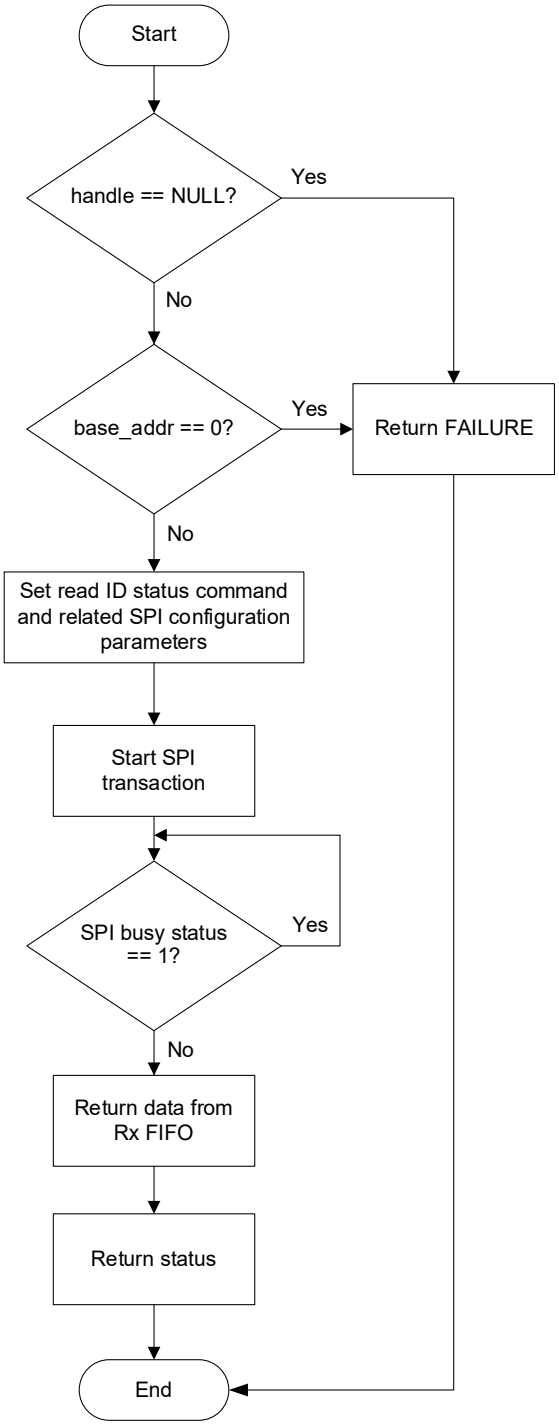

Figure 3.4. unsigned char gencmd_flash_rdid()

3.5. gencmd_flash_wrsr123()

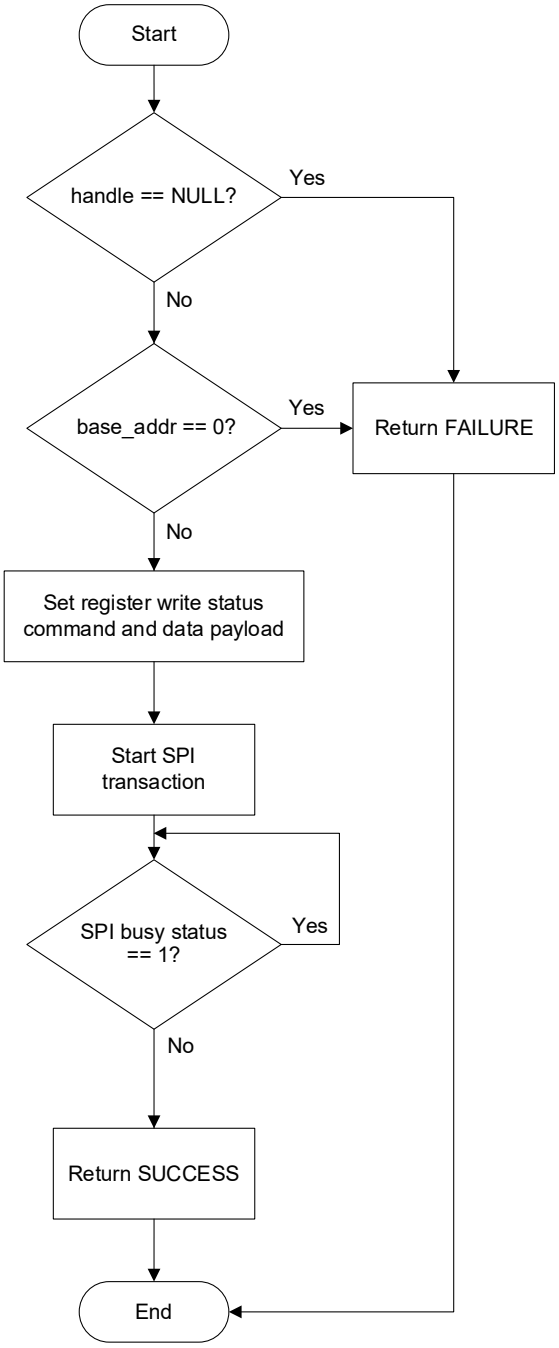

Figure 3.5. unsigned char gencmd_flash_wrsr123()

3.6. gencmd_flash_rdsr123()

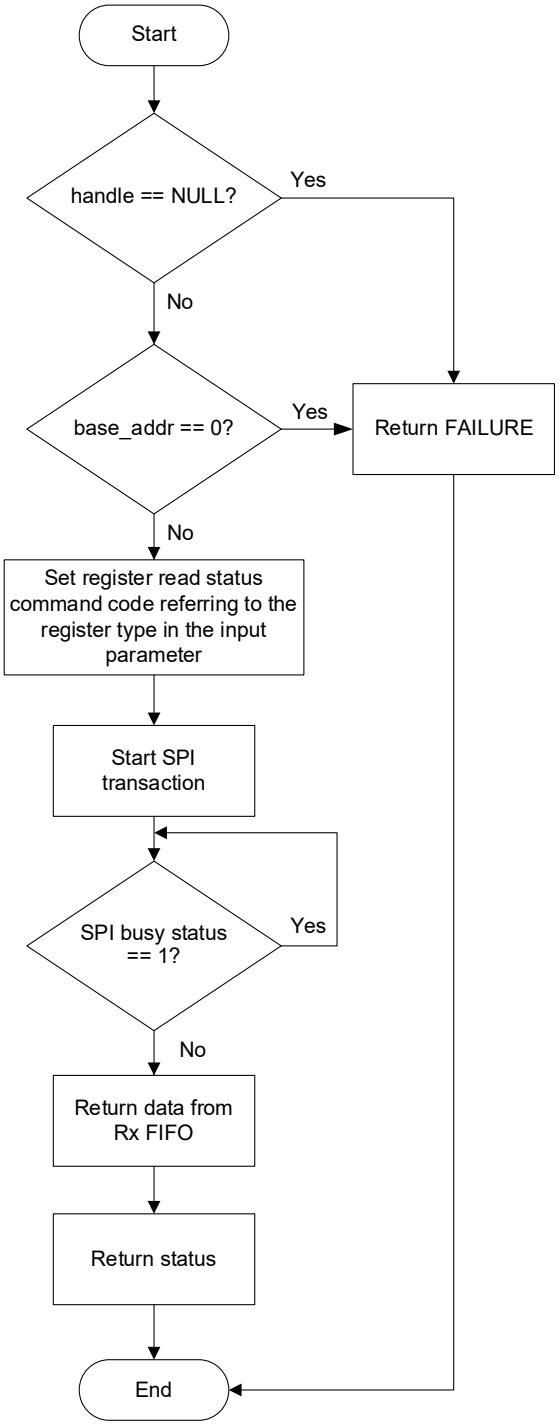

Figure 3.6. unsigned char gencmd_flash_rdsr123()

3.7. gencmd_flash_set_4byte_mode()

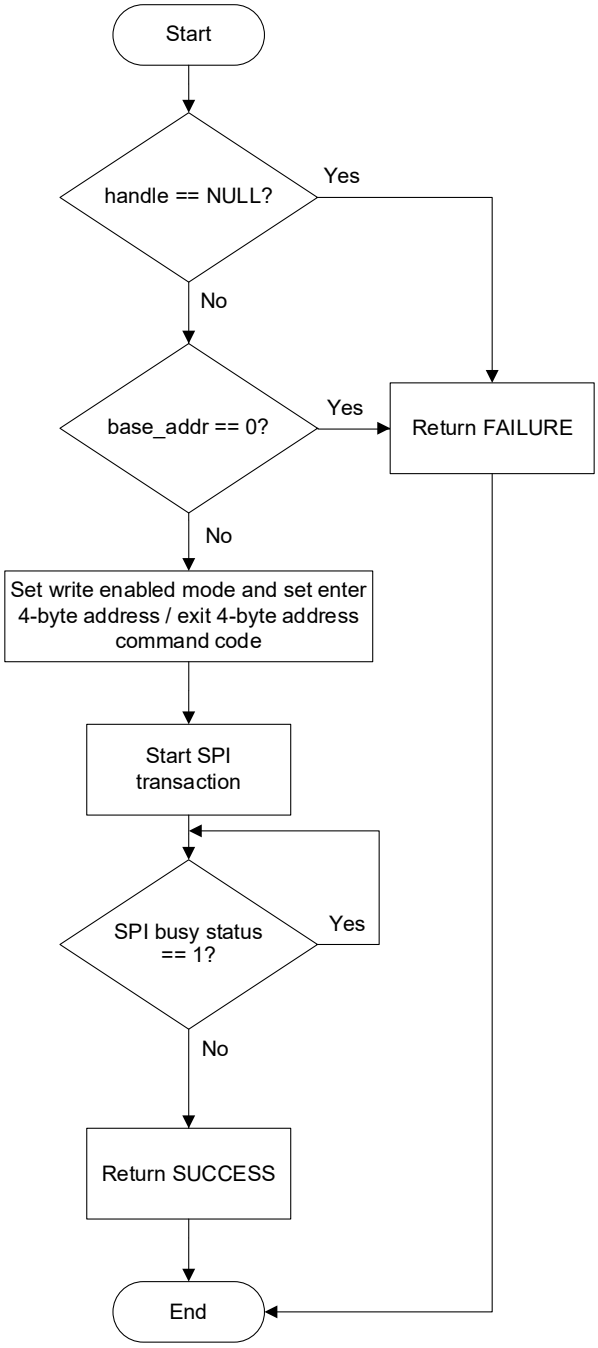

Figure 3.7. unsigned char gencmd_flash_set_4byte_mode()

3.8. gencmd_flash_set_quad_mode()

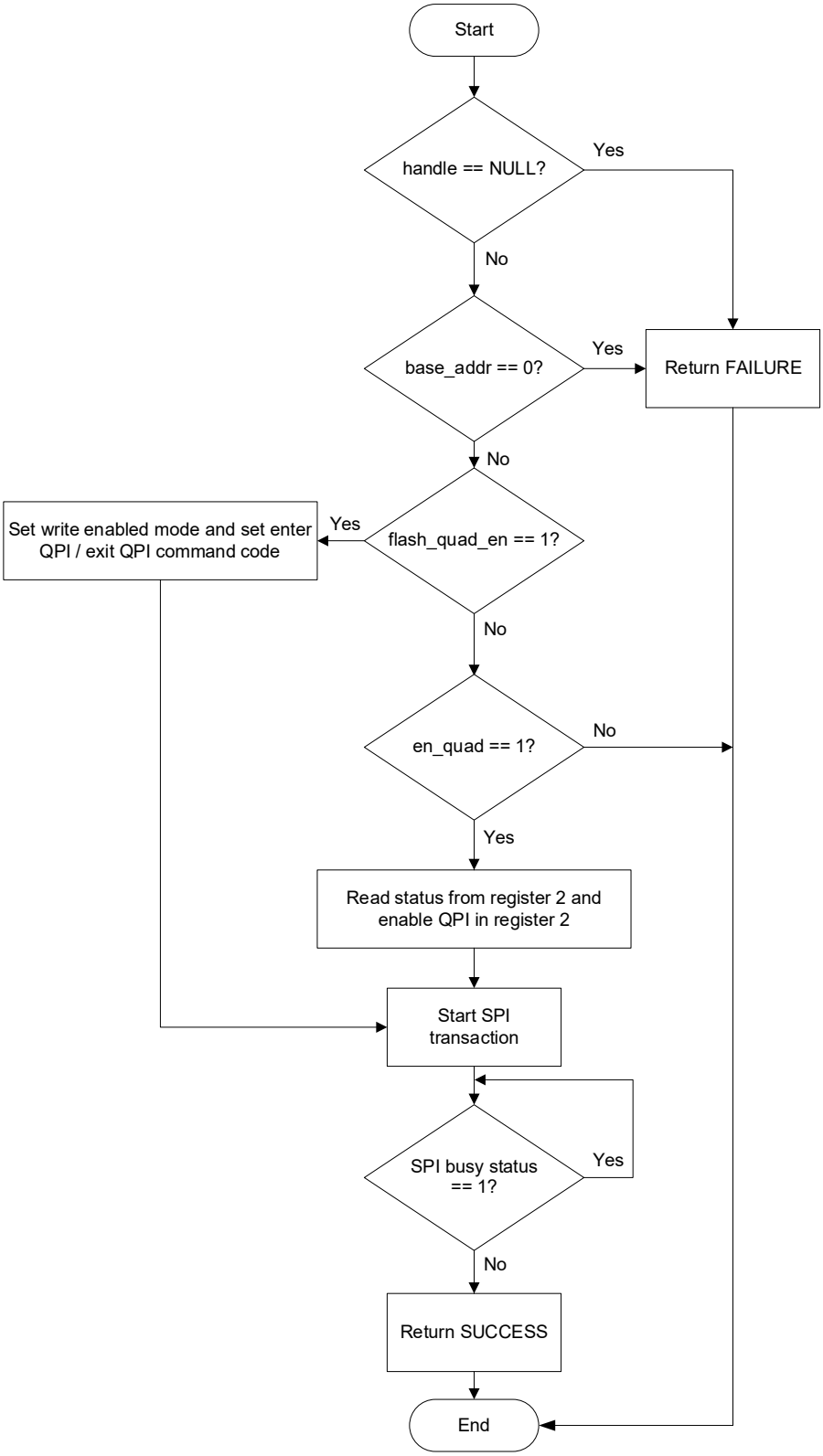

Figure 3.8. unsigned char gencmd_flash_set_quad_mode()

3.9. gencmd_flash_soft_reset()

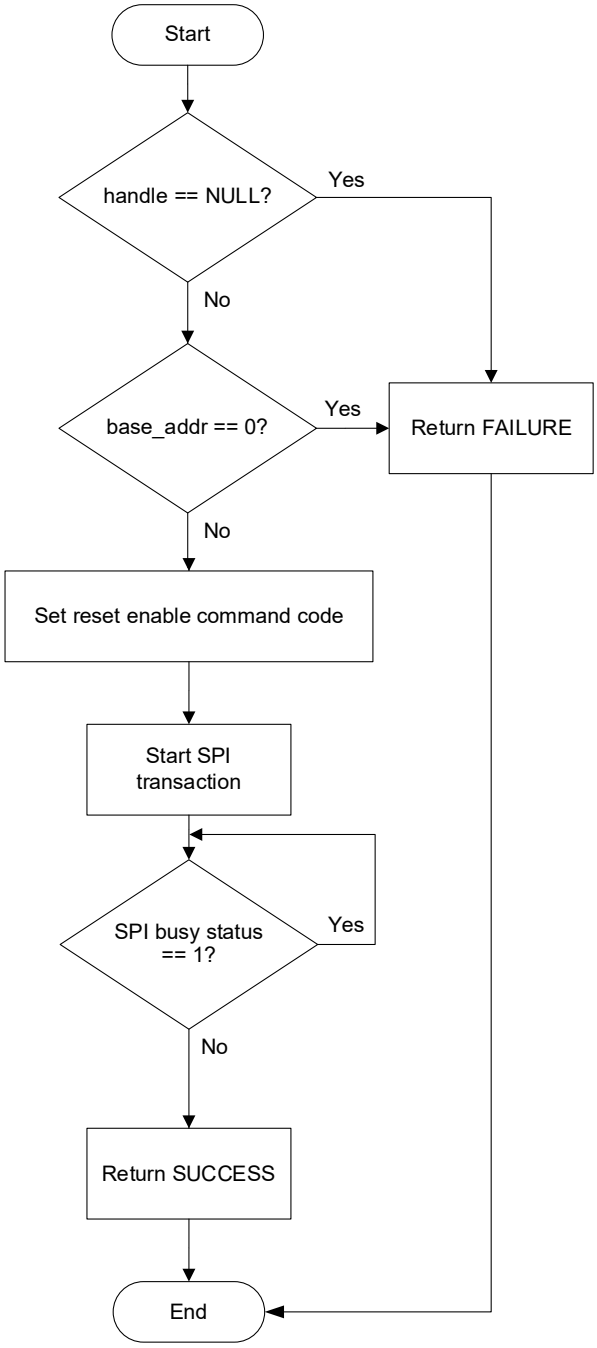

Figure 3.9. unsigned char gencmd_flash_soft_reset()

3.10. gencmd_quadspi_4kb_erase()

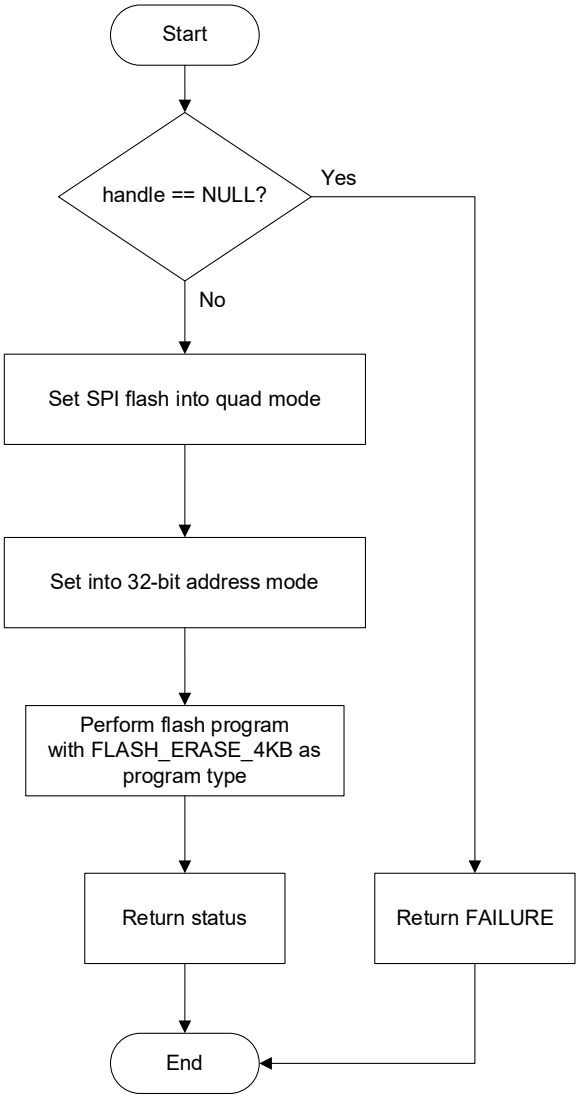

Figure 3.10. unsigned char gencmd_quadspi_4kb_erase()

3.11. gencmd_quadspi_page_program()

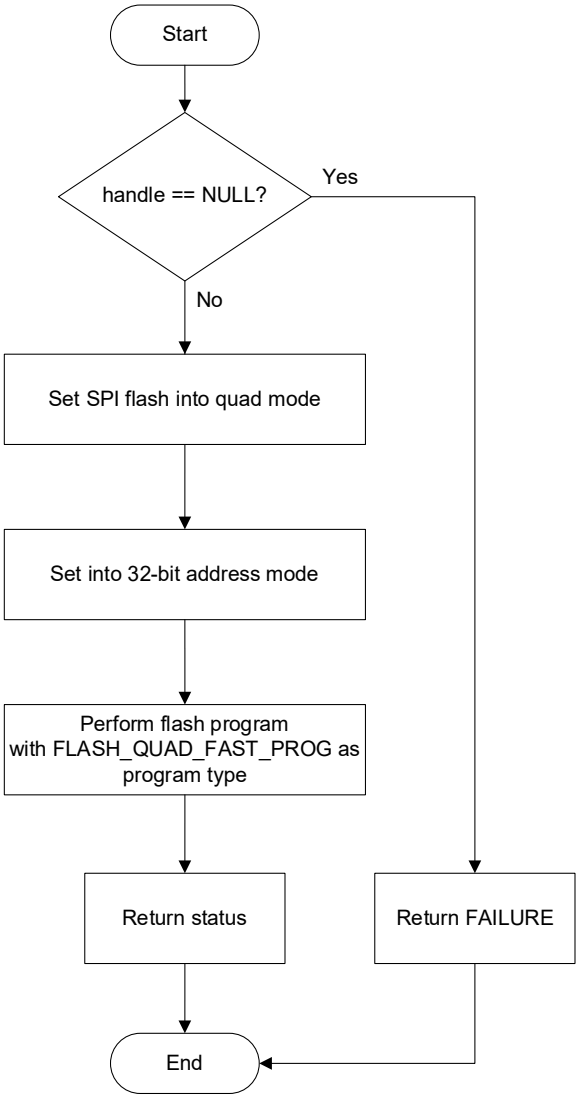

Figure 3.11. unsigned char gencmd_quadspi_page_program()

3.12. config_supported_flash_cmd()

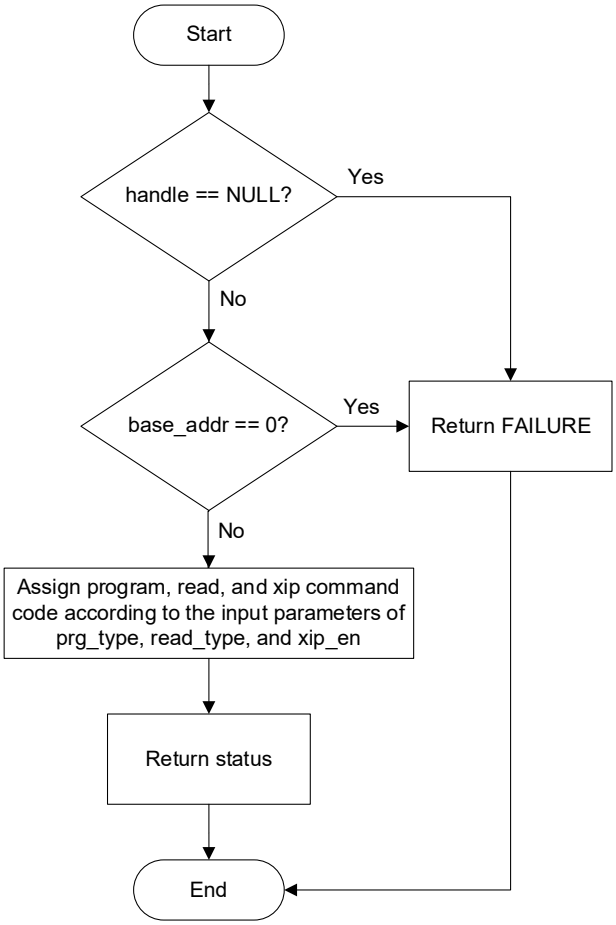

Figure 3.12. unsigned char config_supported_flash_cmd()

3.13. gencmd_flash_rdsr()

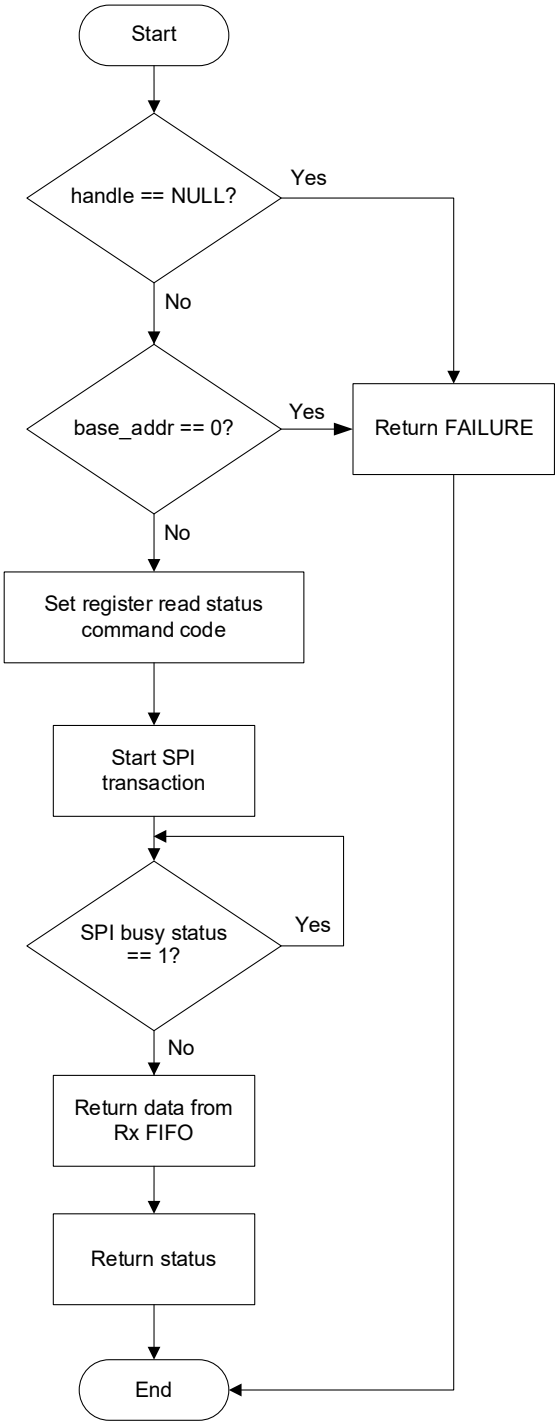

Figure 3.13. unsigned char gencmd_flash_rdsr()

3.14. set_quad_flash_command_config()

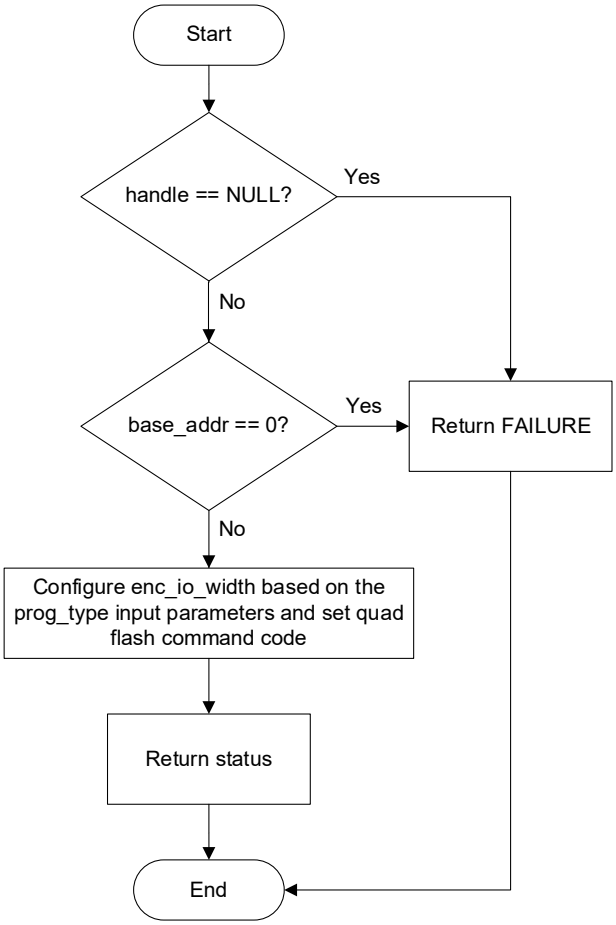

Figure 3.14. unsigned char set_quad_flash_command_config()

3.15. gencmd_quadspi_fast_read()

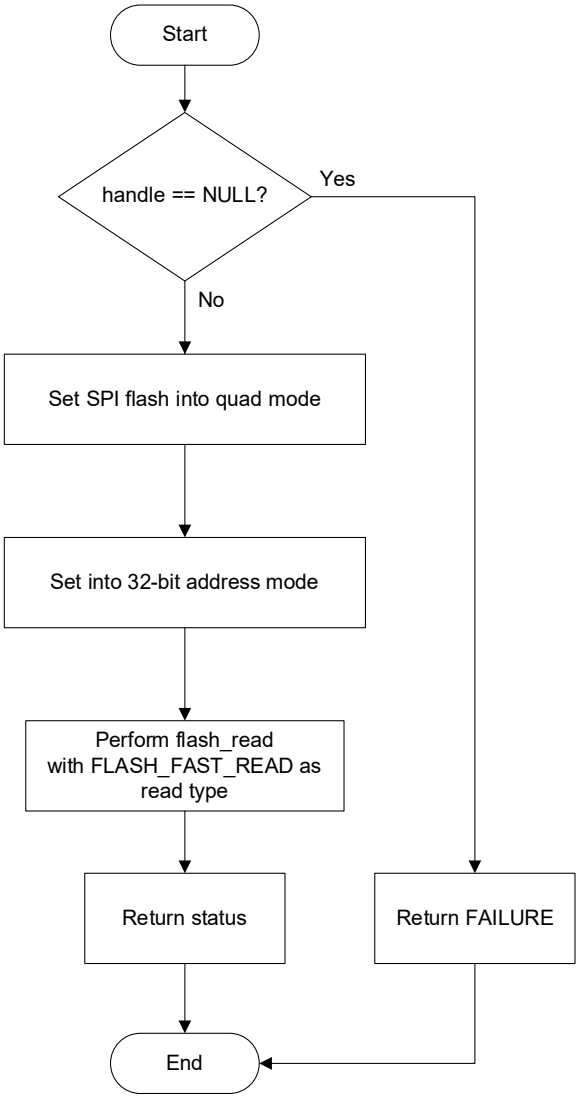

Figure 3.15. unsigned char gencmd_quadspi_fast_read()

3.16. quad_xip_activate()

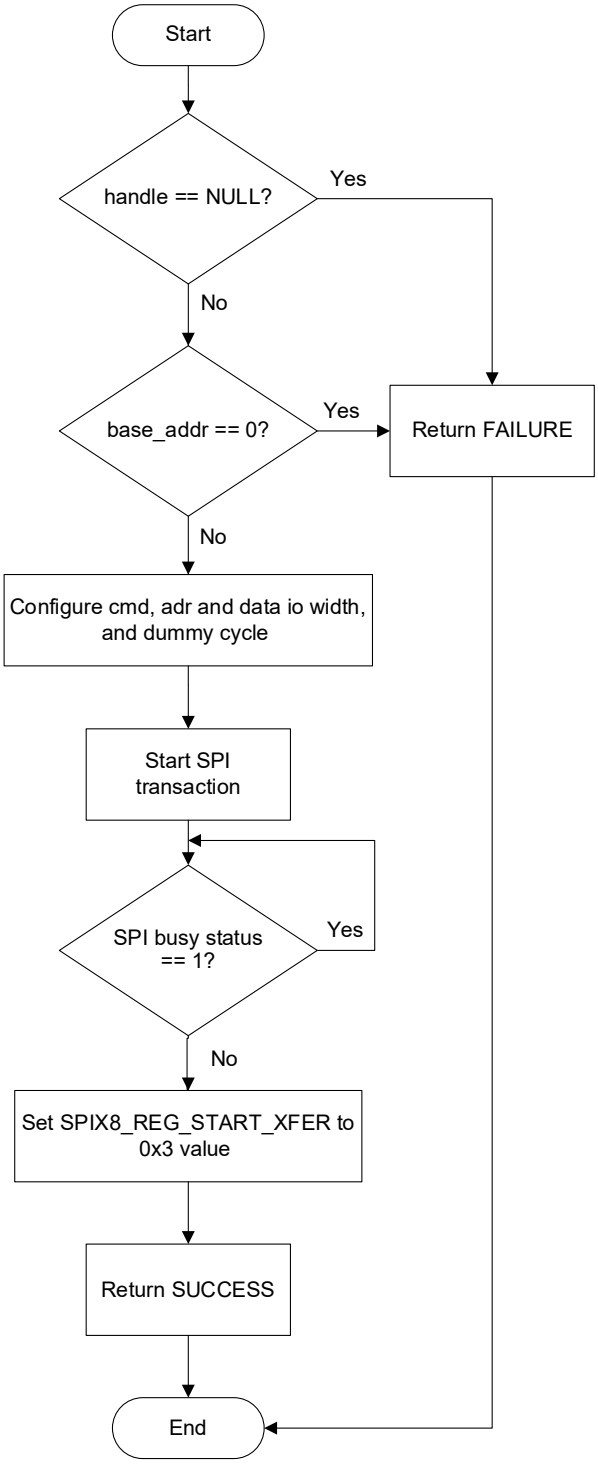

Figure 3.16. unsigned char quad_xip_activate()

3.17. gencmd_wr_vcr_set_octal_mode()

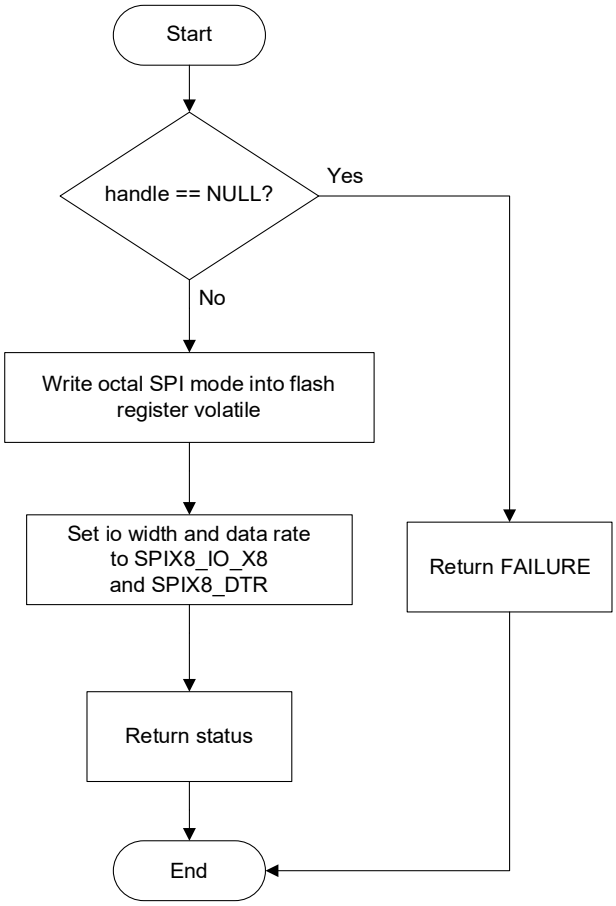

Figure 3.17. uint32_t gencmd_wr_vcr_set_octal_mode()

3.18. gencmd_octspi_erase()

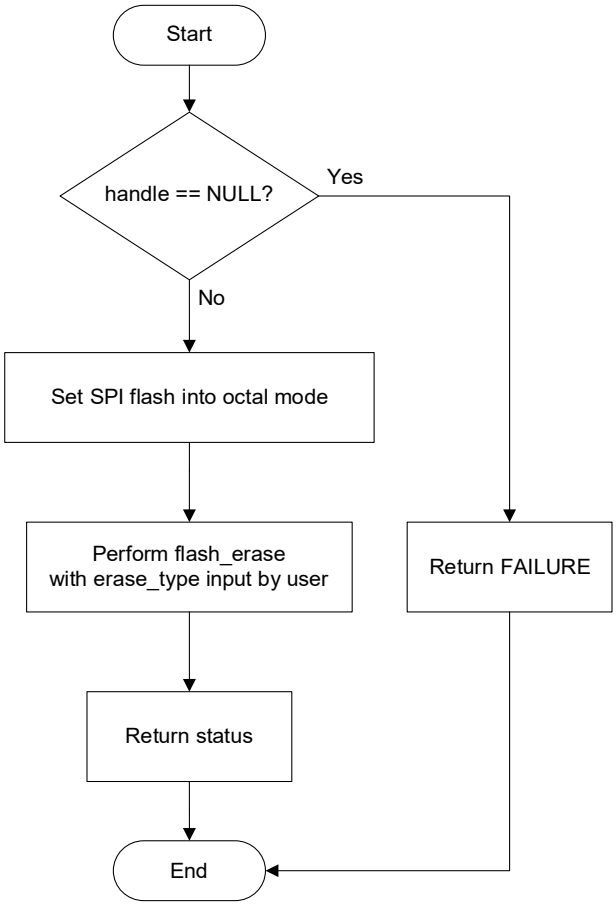

Figure 3.18. unsigned char gencmd_octspi_erase()

3.19. gencmd_octspi_fast_read()

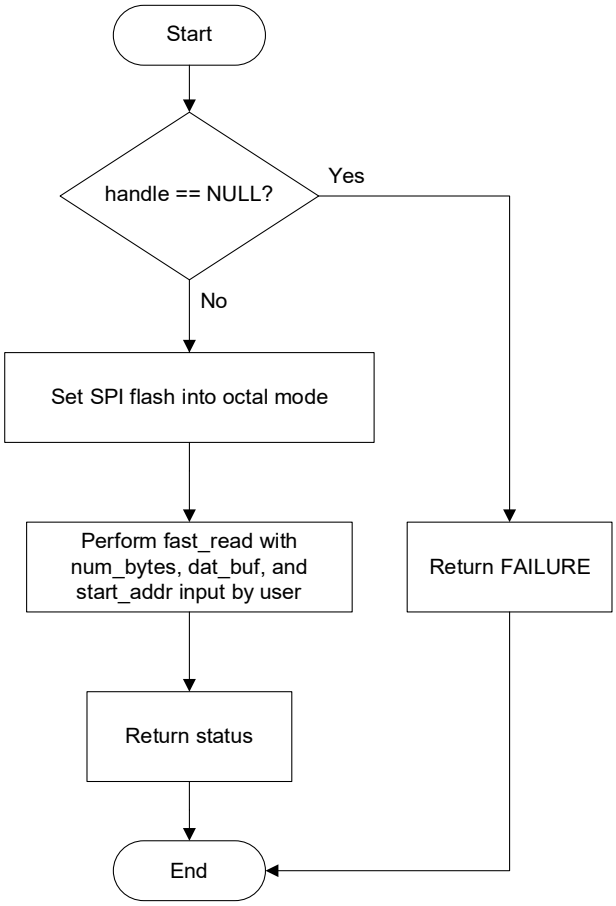

Figure 3.19. unsigned char gencmd_octspi_fast_read()

3.20. gencmd_octspi_page_program()

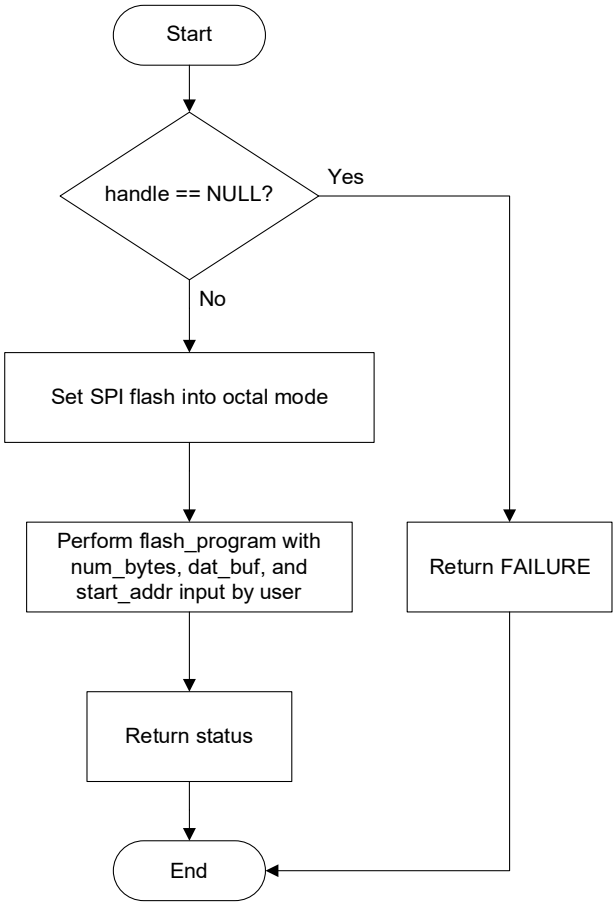

Figure 3.20. unsigned char gencmd_octspi_page_program()

4. API Data Structures

4.1. struct spix8_ctl_reg_t

Table 4.1. spix8_ctl_reg_t Parameters

Data Type	Struct Member	Description
volatile unsigned int	SPIX8_REG_IPCORE_ID	IP core ID
volatile unsigned int	SPIX8_REG_CONFIG0	Configuration 0 register
volatile unsigned int	SPIX8_REG_CONFIG1	Configuration 1 register
volatile unsigned int	SPIX8_REG_SUPCMD_CODE0	Supported command code 0 register
volatile unsigned int	SPIX8_REG_SUPCMD_CODE1	Supported command code 1 register
volatile unsigned int	SPIX8_REG_SUPCMD_CODE2	Supported command code 2 register
volatile unsigned int	SPIX8_REG_SUPCMD_CFG	Supported command configuration register
volatile unsigned int	reserved_00[4]	Reserved bytes
volatile unsigned int	SPIX8_REG_MIN_TGT_SIZE	Minimum target address alignment register
volatile unsigned int	SPIX8_REG_TGT_ADDROFST	Target start address offset register
volatile unsigned int	SPIX8_REG_TOT_TGT_SIZE	Total target address map alignment register
volatile unsigned int	SPIX8_REG_TGT_AXI4BASE	Target AXI4 base address register
volatile unsigned int	SPIX8_REG_INT_ENABLE	Interrupt enable register
volatile unsigned int	reserved_01[48]	Reserved bytes
volatile unsigned int	SPIX8_REG_INT_STATUS	Interrupt status register
volatile unsigned int	SPIX8_REG_GENCMD_CNTR	Generic SPI command packet counter register
volatile unsigned int	SPIX8_REG_SUPCMD_CNTR	Supported flash command packet counter register
volatile unsigned int	SPIX8_REG_DEBUG_INFO0	Octal SPI debug information 0 register
volatile unsigned int	SPIX8_REG_DEBUG_INFO1	Octal SPI debug information 1 register
volatile unsigned int	reserved_02[59]	Reserved bytes
volatile unsigned int	SPIX8_REG_TX_FIFO	Tx FIFO register
volatile unsigned int	SPIX8_REG_RX_FIFO	Rx FIFO register
volatile unsigned int	SPIX8_REG_PKT_HDR0	Packet header 0 register
volatile unsigned int	SPIX8_REG_PKT_HDR1	Packet header 1 register
volatile unsigned int	SPIX8_REG_PKT_HDR2	Packet header 2 register
volatile unsigned int	SPIX8_REG_PKT_HDR3	Packet header 3 register
volatile unsigned int	SPIX8_REG_WRITE_DATA	Write data register
volatile unsigned int	SPIX8_REG_READ_DATA	Read data register
volatile unsigned int	SPIX8_REG_START_XFER	Start transaction register
volatile unsigned int	SPIX8_REG_INT_SET	Register name
volatile unsigned int	SPIX8_REG_TEST_MODE	Interrupt set register
volatile unsigned int	SPIX8_REG_SOFT_RESET	Test mode register
volatile unsigned int	SPIX8_REG_IO_DELAY_CTL	Soft reset register

4.2. struct spix8_ctl_handle_t

Table 4.2. spix8_ctl_handle_t Parameters

Data Type	Struct Member	Description
unsigned int	init_done	Initialization done flag
unsigned int	base_addr	Base address
unsigned int	sck_rate	Clock rate
unsigned int	autoclr_txstart	Auto-clear the tx_start register
unsigned int	autoclr_softrst	Auto-clear the soft reset
unsigned int	spi_io_width	SPI I/O width
unsigned int	max_num_lane	Maximum number of lanes supported
unsigned int	sys_clk_freq	System clock frequency
unsigned int	spi_actual_freq	SPI actual frequency
unsigned int	interrupt_enable	Interrupt enabled flag
unsigned int	spi_mode	SPI mode (standard, dual, quad, octal)
unsigned int	lsb_first	LSB first setting
unsigned int	endianness	Endianness supported (big, small)
unsigned int	en_tgtaddr_map	Enable target address mapping
unsigned int	use_ds_in_ddr	Use data strobe in DDR mode
unsigned int	non_block_txfifo	Use non-blocking Tx FIFO
unsigned int	non_block_rxfifo	Use non-blocking Rx FIFO
unsigned int	spi_dat_rate	SPI data rate
unsigned int	min_tgtaddr_size	Minimum target address size
unsigned int	tot_tgtaddr_size	Total target address size
unsigned int	axi4_tgt_baddr	AXI4 target base address
unsigned int	flash_addr_offset	Flash address offset
unsigned int	flash_quad_en	Flash quad mode enable
unsigned int	flash_addr_mode	Flash address mode
unsigned int	flash_xip_mode	Flash XiP mode
unsigned int	flash_dummy_cycle	Dummy cycle
unsigned int	flash_wrap_cfg	Dummy wrap configuration

5. API Variables

```
spix8_ctl_handle_t octal_spi_c0;
spix8_ctl_handle_t *octal_spi_c0_inst;
const uint32_t flash_addr_offset;
```

These variables are used to access the data members (registers) of the given structure.

6. API Macros

Table 6.1. API Macros Description

Macro	Description
#define GET_REG_MASK(F_LEN,F_IDX)	Gets mask from bytes
#define PUT_FIELD_VAL(F_VALUE,F_LEN,F_IDX)	Inserts bits into bytes
#define SET_REG_FIELD(REG_VAR,F_VALUE,F_LEN,F_IDX)	Sets bits into registers
#define CLEAR	Clears all bits
#define SET_ALL_BITS	Sets all bits
#define DAT32B_SWAP_BYTES(DATA)	Swaps bytes between data endianness (32 bits)
#define DAT24B_SWAP_BYTES(DATA)	Swaps bytes between data endianness (24 bits)
#define GET8B_AND_CONCAT4X(DATA)	Concatenates 8 bits of data into 32 bits
#define GET_FIELD_VAL(DATA, F_IDX, F_LEN)	Reads bits from bytes
#define CALC_DWORD_LEN(BYTE_LEN)	Calculates the DWORD length
#define IP_CORE_RST	Soft reset for IP core
#define IP_CSR_RST	Soft reset for CSR
#define TX_FIFO_RST	Soft reset for Tx FIFO
#define RX_FIFO_RST	Soft reset for Rx FIFO
#define SPI_TGT_RST	Soft reset for SPI target
#define SPIX8_INT_FLASH_PROGRAM_FAIL	SPI flash program fail interrupt
#define SPIX8_INT_FLASH_ERASE_FAIL	SPI flash erase fail interrupt
#define SPIX8_INT_RD_ON_EMTY_ERROR	SPI read buffer empty error interrupt
#define SPIX8_INT_WR_ON_FULL_ERROR	SPI write buffer full error interrupt
#define SPIX8_INT_BUS_ACCESS_ERROR	SPI bus access error interrupt
#define SPIX8_INT_USER_PKT_DECODE_ERROR	User packet decode error interrupt
#define SPIX8_INT_SUP_RD_TRANS_CNT_HIT	Supported read transmission count threshold hit interrupt
#define SPIX8_INT_SUP_WR_TRANS_CNT_HIT	Supported write transmission count threshold hit interrupt
#define SPIX8_INT_GEN_RD_TRANS_CNT_HIT	Generic read transmission count threshold hit interrupt
#define SPIX8_INT_GEN_WR_TRANS_CNT_HIT	Generic write transmission count threshold hit interrupt
#define SPIX8_INT_RDATA_AVAILABLE	Read data available interrupt
#define SPIX8_INT_WDATA_DONE	Write data done interrupt
#define SPIX8_INT_RXFIFO_NOT_EMTY	Rx FIFO not empty interrupt
#define SPIX8_INT_RXFIFO_AFUL	Rx FIFO full interrupt
#define SPIX8_INT_TXFIFO_EMTY	Tx FIFO empty interrupt
#define SPIX8_INT_TXFIFO_AFUL	Tx FIFO full interrupt
#define SPIX8_CONFIG0_SCK_RATE	SPI configuration 0 clock rate
#define SPIX8_CONFIG0_AUTO_CLR_RST	SPI configuration 0 auto clear reset
#define FLASH_CMDPATTERN_0	SPI flash command pattern 0
#define FLASH_CMDPATTERN_1	SPI flash command pattern 1
#define FLASH_CMDPATTERN_2	SPI flash command pattern 2
#define FLASH_CMDPATTERN_3	SPI flash command pattern 3
#define USRCMD_REG_CFG_SETTING	User command register configuration settings
#define USRCMD_PKT_HDR_SETTING	User command packet header settings
#define SPIX8_GEN_CMD	SPI generic commands
#define SPIX8_SUP_CMD	SPI supported commands
#define SPIX8_WR_CMD	SPI write commands
#define SPIX8_RD_CMD	SPI read commands
#define SPIX8_IO_X1	SPI I/O x1 lanes
#define SPIX8_IO_X2	SPI I/O x2 lanes

Macro	Description
#define SPIX8_IO_X4	SPI I/O x4 lanes
#define SPIX8_IO_X8	SPI I/O x8 lanes
#define SPIX8_STR	Single transfer rate
#define SPIX8_DTR	Double transfer rate

References

- Octal SPI Controller IP User Guide (FPGA-IPUG-02273)
- Octal SPI Controller IP Release Notes (FPGA-RN-02011)
- Avant-E web page
- Avant-G web page
- Avant-X web page
- Certus-NX web page
- CertusPro-NX web page
- Octal SPI 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 1.2, December 2025

Section	Change Summary
Introduction	• Updated device families and devices in the Audience section. • Updated driver version in the Driver Version section. • Updated driver version and IP version in the Driver and IP Compatibility section.
References	Removed CrossLink-NX and MachXO5-NX web pages.

Revision 1.1, June 2025

Section	Change Summary
Introduction	• Updated driver version in the Driver Version section. • Updated driver version and IP version in the Driver and IP Compatibility section. • Added the Flash Device Selection section.
References	• Updated listing name to Lattice Radiant Software web page. • Added the Avant-E, Avant-G, Certus-NX, CrossLink-NX, MachXO5-NX, Octal SPI Controller IP Core, Lattice Propel Design Environment, and Lattice Solutions Reference Designs web pages.

Revision 1.0, February 2025

Section	Change Summary
All	Initial release.

www.latticesemi.com