Internal Flash Controller Driver API Reference

Technical Note

FPGA-TN-02421-1.0

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
AHB-L	Advanced High-Performance Bus Lite
APB	Advanced Peripheral Bus
API	Application Programming Interface
CFG	Configuration
FIFO	First In, First Out
IFC	Internal Flash Controller
IP	Intellectual Property
RD	Read
UFM	User Flash Memory
WR	Write

1. Introduction

The Internal Flash Controller IP for MachXO5™-NX enables access to the internal flash memory of the MachXO5-NX device using the Advanced High-Performance Bus Lite (AHB-L) or Advanced Peripheral Bus (APB) interface.
Refer to the Internal Flash Controller IP for MachXO5-NX User Guide (FPGA-IPUG-02174) 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 MachXO5-NX devices. The technical guide assumes readers have expertise in embedded systems and FPGA technologies.

1.3. Driver Version

INTERNAL_FLASH_CONTROLLER_DRV_VER “25.2.0”

1.4. Driver and IP Compatibility

Driver version	IP version
25.2.0	2.3.0

Refer to the Internal Flash Controller IP for MachXO5-NX Release Notes (FPGA-RN-02085) for more information on the driver and IP versions.

2. API Description

2.1. ifc_status_get()

This API is used to fetch the status of a given internal flash controller (IFC) instance and store it at the memory location indicated by the status pointer.

```
unsigned int ifc_status_get(struct ifc_instance *this_ifc, unsigned char *status)
```

In/Out	Parameter	Description	Returns
In	*this_ifc	Handle of the struct ifc_instance.	0: success 1: failure
Out	*status	Variable address where the status value is stored.	

2.2. ifc_int_en()

This API is used to set the interrupt enable register of a given IFC instance to the specified value.

```
unsigned int ifc_int_en(struct ifc_instance *this_ifc, unsigned char int_enable)
```

In/Out	Parameter	Description	Returns
In	*this_ifc	Handle of the struct ifc_instance.	0: success 1: failure
In	int_enable	**Note:** For each bit, set to 1 to enable or set to 0 to disable. Bit 0: Enable/Disable RD FIFO empty interrupt Bit 1: Enable/Disable RD FIFO almost empty interrupt Bit 2: Enable/Disable RD FIFO almost full interrupt Bit 3: Enable/Disable RD FIFO full interrupt Bit 4: Enable/Disable WR FIFO empty interrupt Bit 5: Enable/Disable WR FIFO almost empty interrupt Bit 6: Enable/Disable WR FIFO almost full interrupt Bit 7: Enable/Disable WR FIFO full interrupt Bit 31: Enable/Disable command done interrupt	

2.3. ifc_int_set()

This API is used to set the interrupt set register of a given IFC instance to the specified value.

```
unsigned int ifc_int_set(struct ifc_instance *this_ifc, unsigned char int_set_val)
```

In/Out	Parameter	Description	Returns
In	*this_ifc	Handle of the struct ifc_instance.	0: success 1: failure
In	int_set_val	**Note:** Write 1 to force assert the corresponding interrupt. Bit 0: Set RD FIFO empty interrupt Bit 1: Set RD FIFO almost empty interrupt Bit 2: Set RD FIFO almost full interrupt Bit 3: Set RD FIFO full interrupt Bit 4: Set WR FIFO empty interrupt Bit 5: Set WR FIFO almost empty interrupt Bit 6: Set WR FIFO almost full interrupt Bit 7: Set WR FIFO full interrupt Bit 31: Set command done interrupt	

2.4. ifc_init()

This API is used to initialize an IFC instance with the provided parameters.

```
unsigned int ifc_init(struct ifc_instance *this_ifc, unsigned int base_address, unsigned int *memory_map, unsigned int wr_fifo_depth)
```

In/Out	Parameter	Description	Returns
In	*this_ifc	Handle of the struct ifc_instance.	0: success 1: failure
In	base_address	Base address of the IFC IP.	
In	*memory_map	Pointer to an array containing memory map information.	
In	wr_fifo_depth	Write FIFO depth which must not exceed FIFO_DEPTH_64 (64 DWORD).	

2.5. ifc_page_write()

This API is used to write a specified number of words from a buffer to a specific address in the internal flash memory.

```
unsigned int ifc_page_write(struct ifc_instance *this_ifc, unsigned int partition, unsigned int *wr_buf, unsigned int length, unsigned int addr)
```

In/Out	Parameter	Description	Returns
In	*this_ifc	Handle of the struct ifc_instance.	0: success 1: failure
In	partition	Partition to write to must not exceed eight USERDATA partitions.	
In	*wr_buff	Pointer to the buffer containing the data to be written.	
In	length	Length of the data in bytes.	
In	addr	Address within the partition where the data will be written.	

2.6. ifc_block_erase()

This API is used to erase a block of memory in the internal flash controller.

```
unsigned int ifc_block_erase(struct ifc_instance *this_ifc, unsigned int partition,
unsigned int erase_type, unsigned int addr)
```

In/Out	Parameter	Description	Returns
In	*this_ifc	Handle of the struct ifc_instance.	0: success 1: failure
In	partition	Partition to write to must not exceed eight USERDATA partitions.	
In	erase_type	Type of erase operation to perform. 0 – 64k erase 1 – Chip erase	
In	addr	Address within the partition where the data will be erased.	

2.7. ifc_page_read()

This API is used to read a specified number of bytes from a given partition and address in the internal flash memory.

```
unsigned int ifc_page_read(struct ifc_instance *this_ifc, unsigned int partition,
unsigned int *rd_buf, unsigned int length, unsigned int addr)
```

In/Out	Parameter	Description	Returns
In	*this_ifc	Handle of the struct ifc_instance.	0: success 1: failure
In	partition	Partition to write to must not exceed eight USERDATA partitions.	
In	*rd_buf	Pointer to the buffer where the read data will be stored.	
In	length	Number of bytes to read.	
In	addr	Starting address within the partition to read from.	

2.8. ifc_soft_reset()

This API is used to perform a soft reset on the specified IFC instance.

```
unsigned int ifc_soft_reset(struct ifc_instance * this_ifc)
```

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

3. Function Call Flow Diagrams

3.1. ifc_status_get()

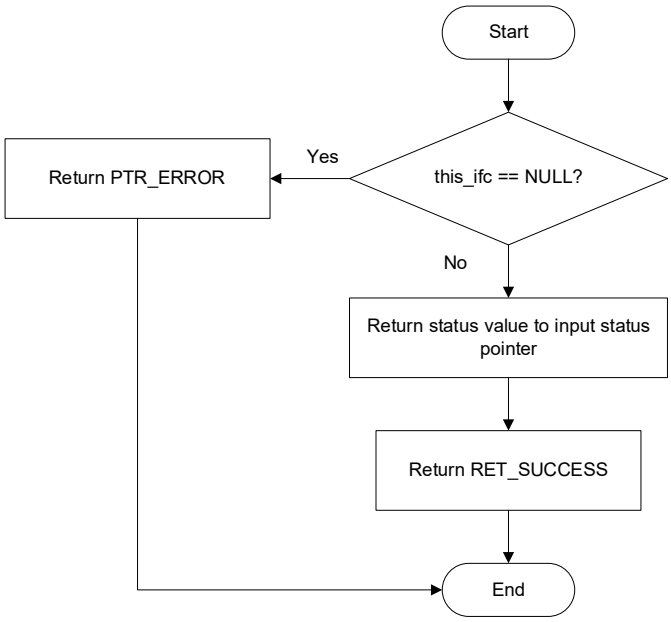

Figure 3.1. unsigned int ifc_status_get()

3.2. ifc_int_en()

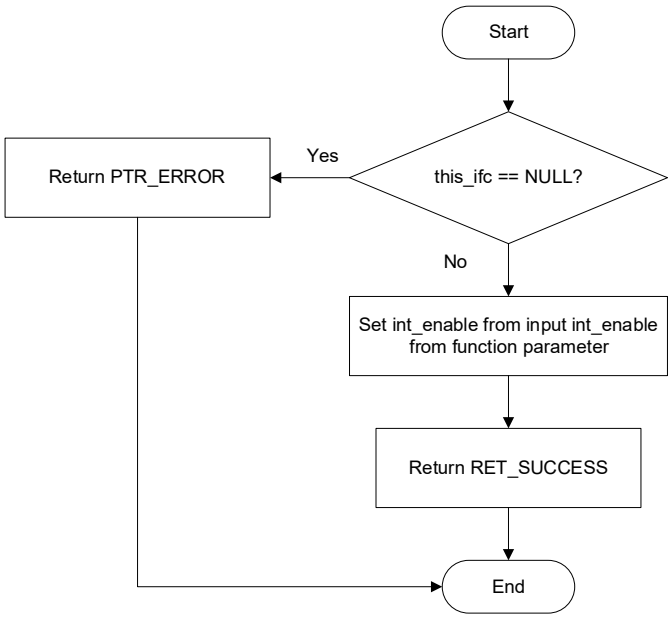

Figure 3.2. unsigned int ifc_int_en()

3.3. ifc_int_set()

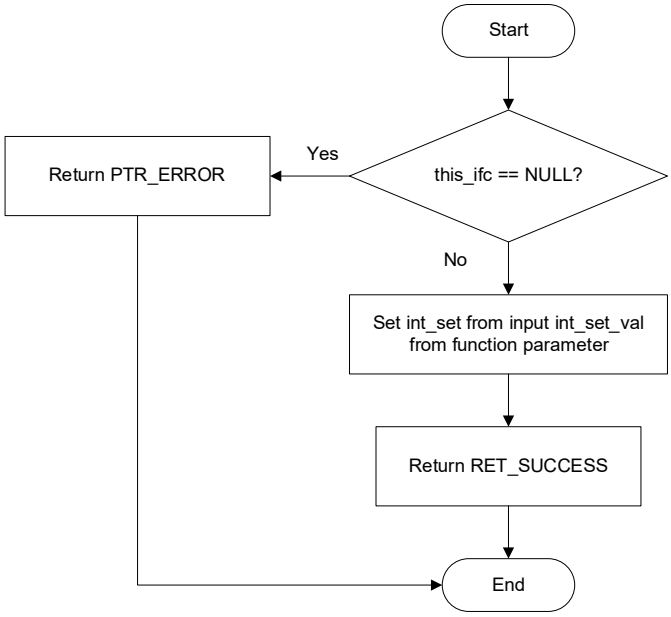

Figure 3.3. unsigned int ifc_int_set()

3.4. ifc_init()

Figure 3.4. unsigned int ifc_init()

3.5. ifc_page_write()

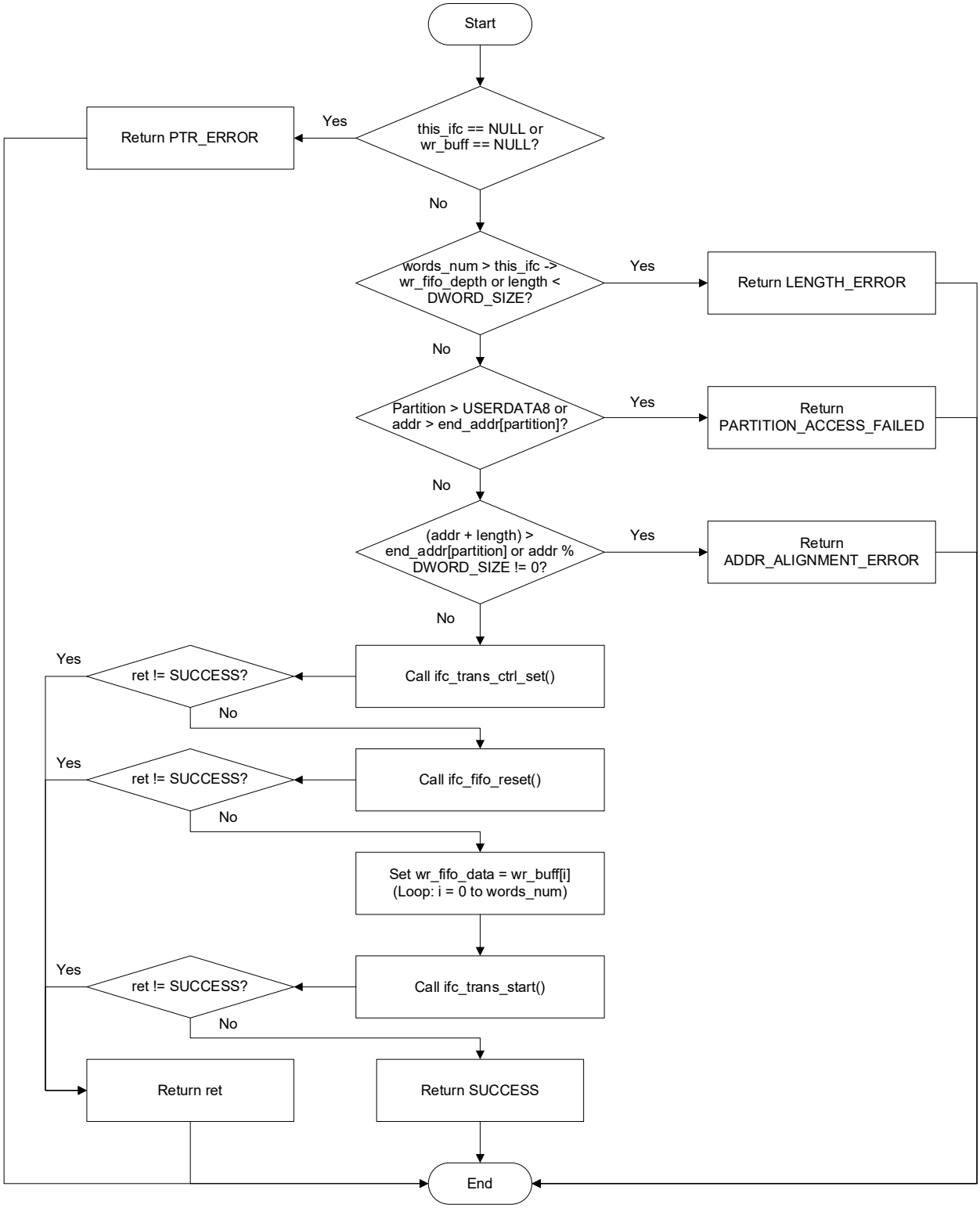

Figure 3.5. unsigned int ifc_page_write()

3.6. ifc_block_erase()

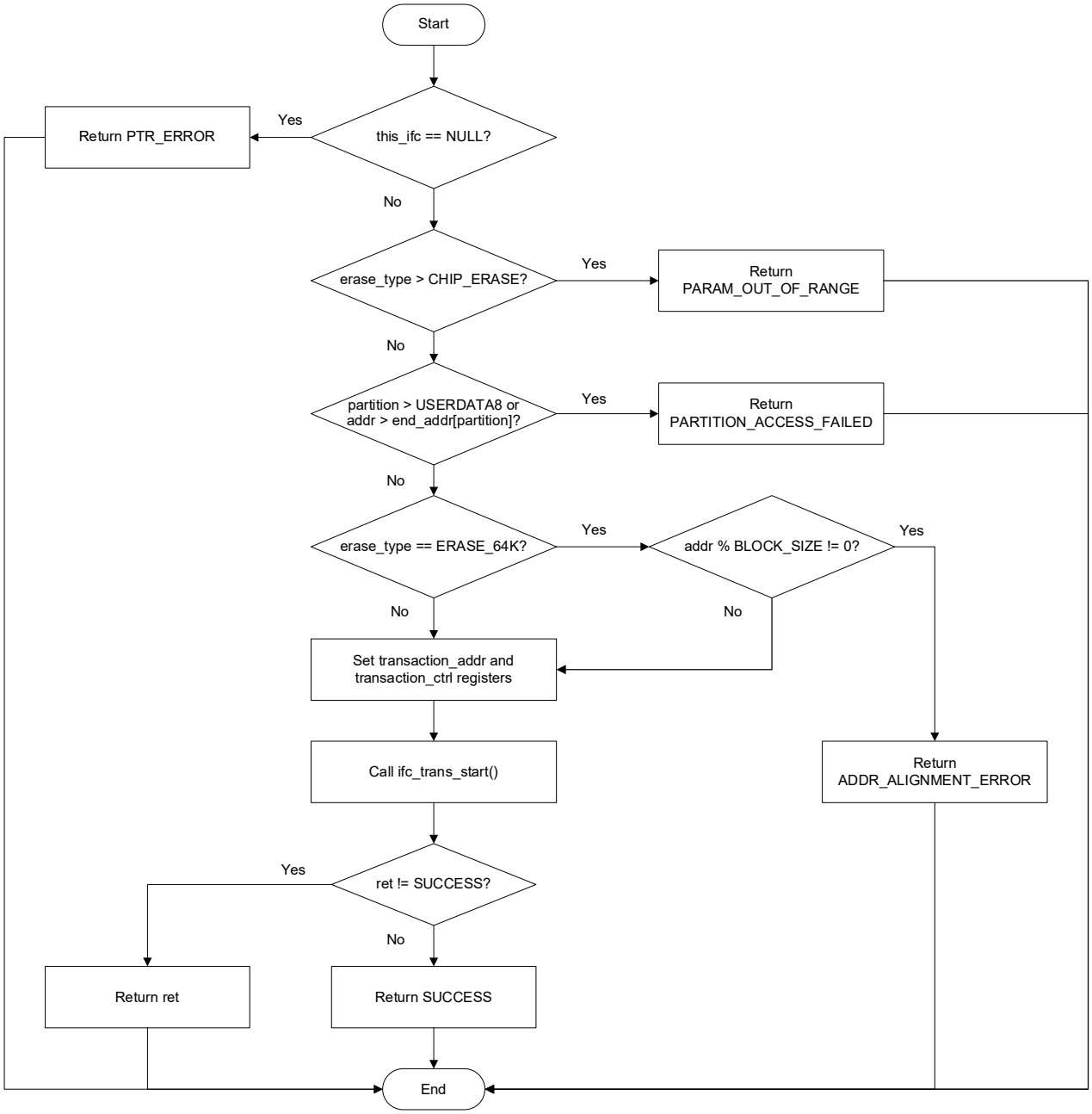

Figure 3.6. unsigned int ifc_block_erase()

3.7. ifc_page_read()

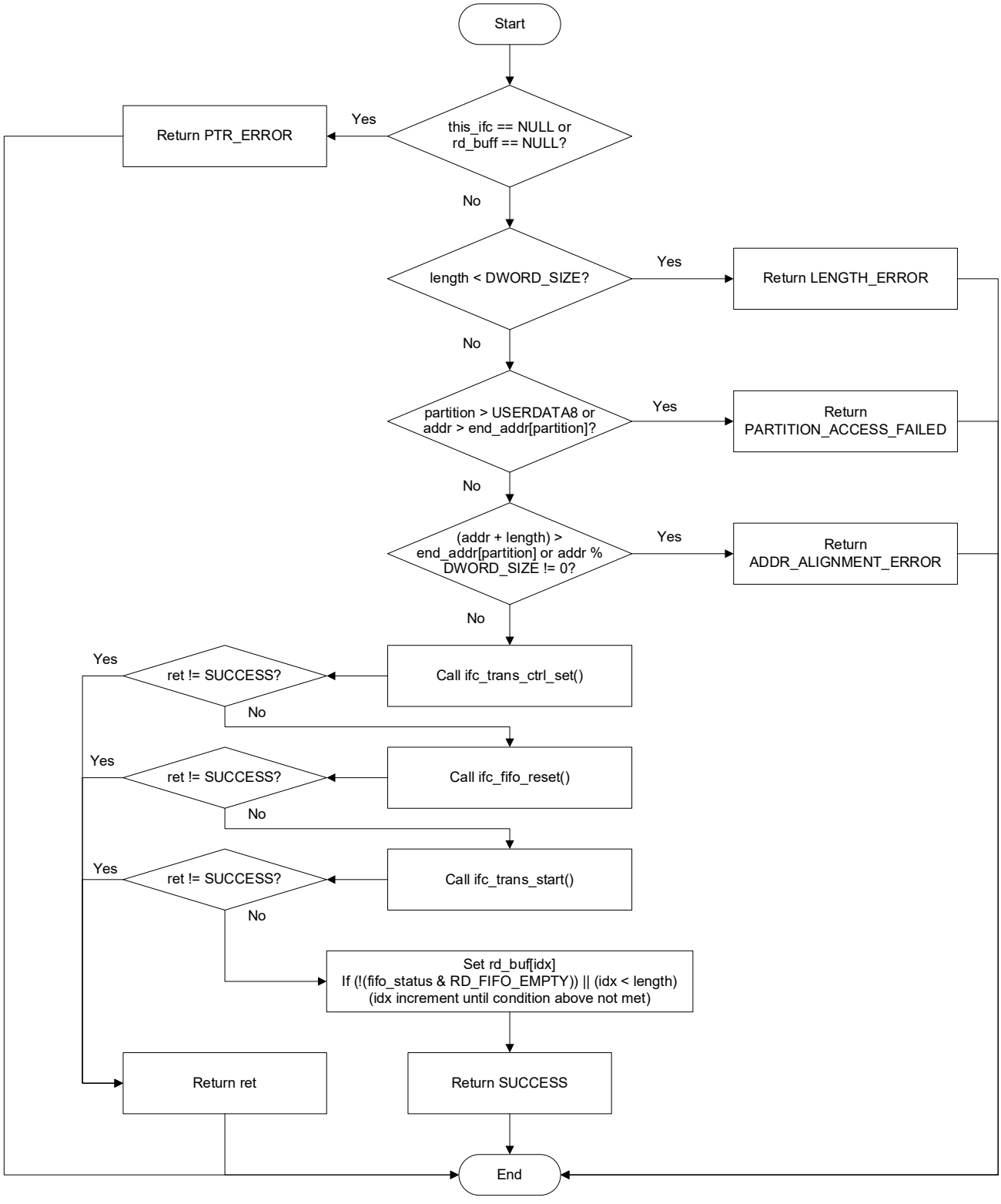

Figure 3.7. unsigned int ifc_page_read()

3.8. ifc_soft_reset()

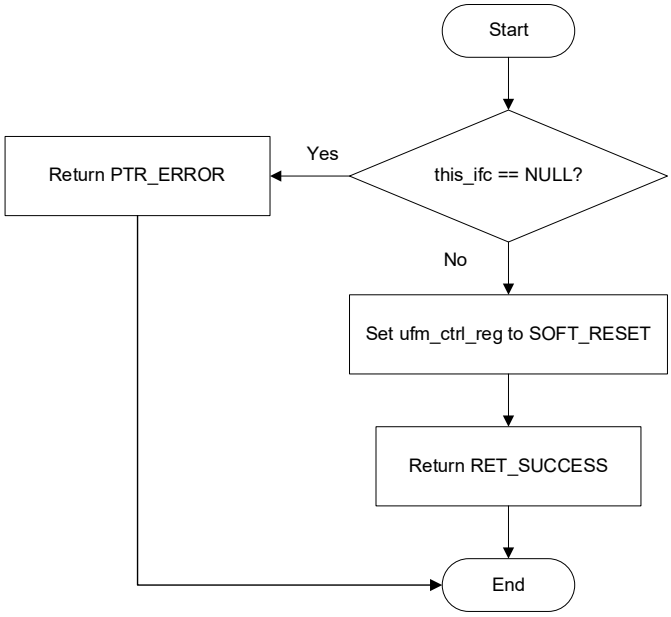

Figure 3.8. unsigned int ifc_soft_reset()

4. API Data Structures

4.1. struct ifc_instance

Table 4.1. ifc_instance Parameters

Data Type	Struct Member	Description
unsigned int	base_address	Base address of the flash controller
unsigned int	cfg_size[MAX_CFG_NUM]	Array containing the individual sizes of CFG1, CFG2, and CFG3
unsigned int	ufm_size[MAX_UFM_NUM]	Array containing the individual sizes of UFM1, UFM2, and UFM3
unsigned int	userdata_size[MAX_USERDATA_NUM]	Array of USERDATA sizes
unsigned int	end_addr[MAX_PARTITION_NUM]	Array containing the end address of each partition
unsigned int	wr_fifo_depth	WR FIFO depth size
unsigned int	status	Status of ifc (busy, idle)

4.2. struct ifc_reg_type_t

Table 4.2. ifc_reg_type_t Parameters

Data Type	Struct Member	Description
volatile unsigned int	ufm_ctrl_reg	IFC register offset 0x0
volatile unsigned int	transaction_addr	IFC register offset 0x4
volatile unsigned int	wr_fifo_data	IFC register offset 0x8
volatile unsigned int	rd_fifo_data	IFC register offset 0xC
volatile unsigned int	transaction_ctrl	IFC register offset 0x10
volatile unsigned int	fifo_status	IFC register offset 0x14
volatile unsigned int	fifo_ctrl	IFC register offset 0x18
volatile unsigned int	int_status	IFC register offset 0x1C
volatile unsigned int	int_enable	IFC register offset 0x20
volatile unsigned int	int_set	IFC register offset 0x24

5. API Enum

5.1. enum IFC_ErrorCode

Table 5.1. IFC_ErrorCode Variables

Enum Member	Enum Decimal Value
SUCCESS	0
PTR_ERROR	1
CFG_ERROR	2
UFM_ERROR	3
USERDATA_ERROR	4
CFG_UFM_ERROR	5
CFG_ACCESS_FAILED	6
UFM_ACCESS_FAILED	7
PARTITION_ACCESS_FAILED	8
USERDATA_ACCESS_FAILED	9
STATUS_ERROR	10
PAGE_WRITE_ERROR	11
PAGE_READ_ERROR	12
PAGE_NUM_ERROR	13
FIFO_ERROR	14
ADRR_ALIGNMENT_ERROR	15
PARAM_OUT_OF_RANGE	16
LENGTH_ERROR	17

5.2. enum IFC_erase_type

Table 5.2. IFC_erase_type Variables

Enum Member	Enum Decimal Value
ERASE_64K	0
CHIP_ERASE	1

5.3. enum IFC_FIFO_Depth

Table 5.3. IFC_FIFO_Depth

Enum Member	Enum Decimal Value
FIFO_DEPTH_4	4
FIFO_DEPTH_8	8
FIFO_DEPTH_16	16
FIFO_DEPTH_32	32
FIFO_DEPTH_64	64

6. API Macros

Table 6.1. API Macros Description

Macro	Description
#define DWORD_SIZE	Size of DWORD
#define PAGE_SIZE	Size of IFC page
#define MAX_CFG_NUM	Maximum CFG number
#define MAX_UFM_NUM	Maximum UFM number
#define MIN_USERDATA_NUM	Minimum USERDATA number
#define MAX_USERDATA_NUM	Maximum USERDATA number
#define PARTITION_BIT	Bit position of partition in register offset 0x4
#define TRANS_BYTE_M1_BIT	Bit position of transaction type in register offset 0x10
#define ERASE_TYPE_BIT	Bit position of erase type in register offset 0x10
#define BLOCK_SIZE	Size of IFC block
#define TRANS_BYTE_M1	Bit mask for transaction type m1 in register offset 0x10
#define TRANS_TYPE_BIT	Bit position of transaction type m1 in register offset 0x10
#define ERASE_TYPE_WIDTH	Bit mask for erase type m1 in register offset 0x10
#define CFG_SIZE_1	Size of CFG0
#define CFG_SIZE_2	Size of CFG1
#define CFG_SIZE_3	Size of CFG2
#define UFM_SIZE_1	Size of UFM0
#define UFM_SIZE_2	Size of UFM1
#define MAX_CFG_UFM_SIZE	Maximum size of CFG0, CFG1, CFG2, UFM0, and UFM1
#define MAX_USERDATA_SIZE	Maximum size of USERDATA0 through USERDATAN

References

- Internal Flash Controller IP for MachXO5-NX User Guide (FPGA-IPUG-02174)
- Internal Flash Controller IP for MachXO5-NX Release Notes (FPGA-RN-02085)
- MachXO5-NX web page
- Internal Flash Controller IP Core web page
- Lattice Propel Design Environment web page
- Lattice Radiant Software 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, December 2025

Section	Change Summary
All	Initial release.

www.latticesemi.com