SGDMA Driver API Reference

Technical Note

FPGA-TN-02340-1.4

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

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
API	Application Programming Interface
DMA	Direct Memory Access
MM2S	Memory Map to Stream
S2MM	Stream to Memory Map
SGDMA	Scatter-Gather Direct Memory Access

1. Introduction

Scatter gather direct memory access (SGDMA) refers to the ability to collect data from multiple non-contiguous memory locations and then *scatter* the data to different destinations or vice versa, where data from different sources are gathered and written to non-contiguous memory locations. SGDMA can be used to receive input or store outputs in the memory.

For more information on the SGDMA IP, refer to the SGDMA Controller IP Core - Lattice Radiant Software User Guide (FPGA-IPUG-02131).

1.1. Purpose

This document is a reference guide for developers that provides details of the C language driver APIs, function calls, and flow charts.

1.2. Audience

The intended audience for this document includes embedded system designers and embedded software developers using the Lattice Semiconductor devices. For details on the specific devices used, refer to the SGDMA Controller IP Core User Guide (FPGA-IPUG-02131).

1.3. Driver Version

SGDMA version 25.01.00.

1.4. Driver and IP Compatibility

Table 1.1. Driver and IP Compatibility

Driver Version	IP Version
25.01.00	2.5.0

Table 1.2. Quick Facts of Driver Tested Environment

Driver Tested on HW Devices	Tool Version
Avant-X Versa Board (LAV-AT-X70ES) CertusPro-NX Versa Board (LFCPNX-100-9LFG672I)	Lattice Propel™ Builder 2025.1
	Lattice Propel SDK 2025.1
	Lattice Radiant™ Software 2025.1
	Radiant Programmer 2025.1

Refer to the SDGMA Controller IP Release Notes (FPGA-RN-02058) for more information on the driver and IP versions.

2. API Description

This section describes the APIs for the SGDMA IP. The driver supports multiple instances of SGDMA IPs within the system. You must initialize additional copies of the sgdma_instance_t variables for each instance.

2.1. sgdma_init()

This API initializes the SGDMA IP base address.

```
unsigned int sgdma_init(sgdma_instance_t *this_sgdma, unsigned int base_addr)
```

Table 2.1. sgdma_init()

In/Out	Parameter	Description	Returns
In/out	*this_sgdma	This structure is initialized with the SGDMA IP address, num_of_desc, and the information of the S2MM and MM2S buffer descriptor addresses.	1: success 0: failure
In	base_addr	Base address that specifies the SGDMA IP base address.	

2.2. sgdma_reset_mm2s()

This API resets the MM2S DMA core.

```
unsigned int sgdma_reset_mm2s(sgdma_instance_t *this_sgdma)
```

Table 2.2. sgdma_reset_mm2s()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA IP base address.	1: success 0: failure

2.3. sgdma_reset_s2mm()

This API resets the S2MM DMA core.

```
unsigned int sgdma_reset_mm2s(sgdma_instance_t *this_sgdma)
```

Table 2.3. sgdma_reset_s2mm()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA IP base address.	1: success 0: failure

2.4. sgdma_reset()

This API resets the MM2S and S2MM DMA core.

```
unsigned int sgdma_reset(sgdma_instance_t *this_sgdma)
```

Table 2.4. sgdma_reset()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA IP base address.	1: success 0: failure

2.5. sgdma_irq_mask()

This API masks the MM2S and S2MM interrupt event upon transfer completion.

```
unsigned int sgdma_irq_mask(sgdma_instance_t *this_sgdma, control_type_t sgdma_cntl_type,
irq_mask_t mask_value)
```

Table 2.5. sgdma_irq_mask()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA IP base address.	1: success 0: failure
In	sgdma_cntl_type	Describes the MM2S control offset and S2MM control offset.	
In	mask_value	Describes the mask and unmask values.	

2.6. get_sgdma_s2mm_reg_status()

This API returns the S2MM status register.

```
unsigned int get_sgdma_s2mm_reg_status(sgdma_instance_t *this_sgdma, sgdma_sts_reg_t *sts_reg)
```

Table 2.6. get_sgdma_s2mm_reg_status()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA IP base address.	1: success 0: failure
Out	*sgdma_sts_type	Returns the S2MM register status.	

2.7. get_sgdma_mm2s_reg_status()

This API returns the MM2S and S2MM status register depending on the sgdma_sts_type offset.

```
unsigned int get_sgdma_mm2s_reg_status(sgdma_instance_t *this_sgdma, sgdma_sts_reg_t *sts_reg)
```

Table 2.7. get_sgdma_mm2s_reg_status()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA IP base address.	1: success 0: failure
Out	*sgdma_sts_type	Returns the MM2S register status.	

2.8. get_mm2s_bd_status()

This API returns the particular MM2S buffer descriptor status field using the index which specifies the number of buffer descriptor.

```
unsigned int get_mm2s_bd_status(sgdma_instance_t *this_sgdma, int idx, unsigned int *status)
```

Table 2.8. get_mm2s_bd_status()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA descriptors address.	1: success 0: failure
In	idx	Index that specifies the MM2S buffer descriptor number in multiple buffer descriptor.	
Out	*status	Returns the status information of a specific MM2S descriptor.	

2.9. get_s2mm_bd_status()

This API returns the particular S2MM buffer descriptor status field using the index that specifies the number of buffer descriptor.

```
unsigned int get_s2mm_bd_status(sgdma_instance_t *this_sgdma, int idx, unsigned int *status)
```

Table 2.9. get_s2mm_bd_status()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA descriptors address.	1: success 0: failure
In	idx	Index that specifies the S2MM buffer descriptor number in multiple buffer descriptor.	
Out	*status	Returns the status information of a specific S2MM descriptor.	

2.10. clear_mm2s_bd_status()

This API clears the status of a specific MM2S descriptor.

```
unsigned int clear_mm2s_bd_status(sgdma_instance_t *this_sgdma, int idx)
```

Table 2.10. clear_mm2s_bd_status()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA IP base address.	1: success 0: failure
In	idx	The index of the MM2S buffer descriptor to clear the status of	

2.11. clear_s2mm_bd_status()

This API clears the status of a specific S2MM descriptor.

```
unsigned int clear_s2mm_bd_status(sgdma_instance_t *this_sgdma, int idx)
```

Table 2.11. clear_s2mm_bd_status()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA IP base address.	1: success 0: failure
In	idx	The index of the S2MM buffer descriptor to clear the status of	

2.12. mm2s_start_transfer()

This API starts the MM2S transfer.

```
unsigned int mm2s_start_transfer(sgdma_instance_t *this_sgdma)
```

Table 2.12. mm2s_start_transfer()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA IP base address.	1: success 0: failure

2.13. mm2s_enable_interrupt()

This API enables the interrupt for MM2S channel in SGDMA instance.

```
unsigned int mm2s_enable_interrupt(sgdma_instance_t *this_sgdma)
```

Table 2.13. mm2s_start_transfer()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA IP base address.	1: success 0: failure

2.14. mm2s_disable_interrupt()

This API disables the interrupt for MM2S channel in SGDMA instance.

```
unsigned int mm2s_disable_interrupt(sgdma_instance_t *this_sgdma)
```

Table 2.14. mm2s_disable_interrupt()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA IP base address.	1: success 0: failure

2.15. mm2s_get_desc_complete_bit()

This API verifies the completion bit and error bit of the mm2s_desc_t descriptor.

```
unsigned int mm2s_get_desc_complete_bit(sgdma_instance_t *this_sgdma, unsigned int idx,
unsigned char *isComplete, unsigned char *isError)
```

Table 2.15. mm2s_get_desc_complete_bit()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Contains information of the base address of MM2S buffer, the length, and the number of descriptors.	1: success 0: failure
In	idx	Index number of descriptors.	
Out	*isComplete	Returns the complete bit information for a specific descriptor.	
Out	*IsError	Returns the error information for a specific descriptor.	

2.16. mm2s_get_all_descs_complete_bit()

This API polls the complete bit of all MM2S descriptors. Note that this API does not have a timeout feature.

```
unsigned int mm2s_get_all_desc_complete_bit(sgdma_instance_t *this_sgdma)
```

Table 2.16. mm2s_get_all_descs_complete_bit()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA IP base address.	1: success 0: failure

2.17. mm2s_buf_desc_dma()

This API configures the single or multiple buffer descriptors depending on the input parameters.

```
unsigned int mm2s_buf_desc_dma(sgdma_instance_t *this_sgdma)
```

Table 2.17. mm2s_buf_desc_dma()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the base address of the MM2S buffer, the length, and the number of descriptors.	1: success 0: failure

2.18. s2mm_start_transfer()

This API starts the S2MM transfer.

```
unsigned int s2mm_start_transfer(sgdma_instance_t *this_sgdma)
```

Table 2.18. s2mm_start_transfer()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA IP base address.	1: success 0: failure

2.19. s2mm_enable_interrupt()

This API enables the interrupt for S2MM channel in SGDMA.

```
unsigned int s2mm_enable_interrupt(sgdma_instance_t *this_sgdma)
```

Table 2.19. s2mm_enable_interrupt()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA IP base address.	1: success 0: failure

2.20. s2mm_disable_interrupt()

This API disables the interrupt for S2MM channel in SGDMA.

```
unsigned int s2mm_enable_interrupt(sgdma_instance_t *this_sgdma)
```

Table 2.20. s2mm_disable_interrupt ()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA IP base address.	1: success 0: failure

2.21. s2mm_get_desc_complete_bit()

This API verifies the completion bit and error bit of the s2mm_desc_t descriptor.

```
unsigned int s2mm_get_desc_complete_bit(sgdma_instance_t *this_sgdma, unsigned int idx,
unsigned char *isComplete, unsigned char *isError)
```

Table 2.21. s2mm_get_desc_complete_bit()

In/Out	Parameter	Description	Returns
In	*this_sgdma	It has information of the base address of S2MM buffer, the length, and the number of descriptors.	1: success 0: failure
In	idx	Index number of descriptors.	
Out	*isComplete	Returns the complete bit information of a specific descriptor.	
Out	*IsError	Returns the error information for a specific descriptor.	

2.22. s2mm_get_all_descs_complete_bit()

This API polls the complete bit of all S2MM descriptors. Note that this API does not have a timeout feature.

```
unsigned int s2mm_get_all_desc_complete_bit(sgdma_instance_t *this_sgdma)
```

Table 2.22. s2mm_get_all_descs_complete_bit()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA IP base address.	1: success 0: failure

2.23. s2mm_buf_desc_dma()

This API configures the single or multiple buffer descriptors depending on the input parameters.

```
unsigned int s2mm_buf_desc_dma(sgdma_instance_t *this_sgdma)
```

Table 2.23. s2mm_buf_desc_dma()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the base address of the MM2S buffer, the length, and the number of descriptors.	1: success 0: failure

2.24. sgdma_register_read()

This API reads data from the MM2S or S2MM register address based on the index value, which is the offset value of the register.

```
unsigned int sgdma_register_read(sgdma_instance_t *this_sgdma, sgdma_reg_type_t index, unsigned
int *reg_data)
```

Table 2.24. sgdma_register_read()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA IP base address.	1: success 0: failure
In	index	Holds the offset value of a particular register.	
Out	*reg_data	Pointer to the value where data is to be stored.	

2.25. sgdma_register_write()

This API writes data from the MM2S or S2MM register address based on the index value, which is the offset value of the register.

```
unsigned int sgdma_register_write(sgdma_instance_t *this_sgdma, sgdma_reg_type_t index,
unsigned int reg_data)
```

Table 2.25. sgdma_register_read()

In/Out	Parameter	Description	Returns
In	*this_sgdma	Holds the SGDMA IP base address.	1: success 0: failure
In	index	Holds the offset value of a particular register.	
In	*reg_data	Holds that data that is written to the register address.	

3. Function Call Flow Diagrams

3.1. sgdma_init()

```
unsigned int sgdma_init(sgdma_instance_t *this_sgdma, unsigned int base_addr)
```

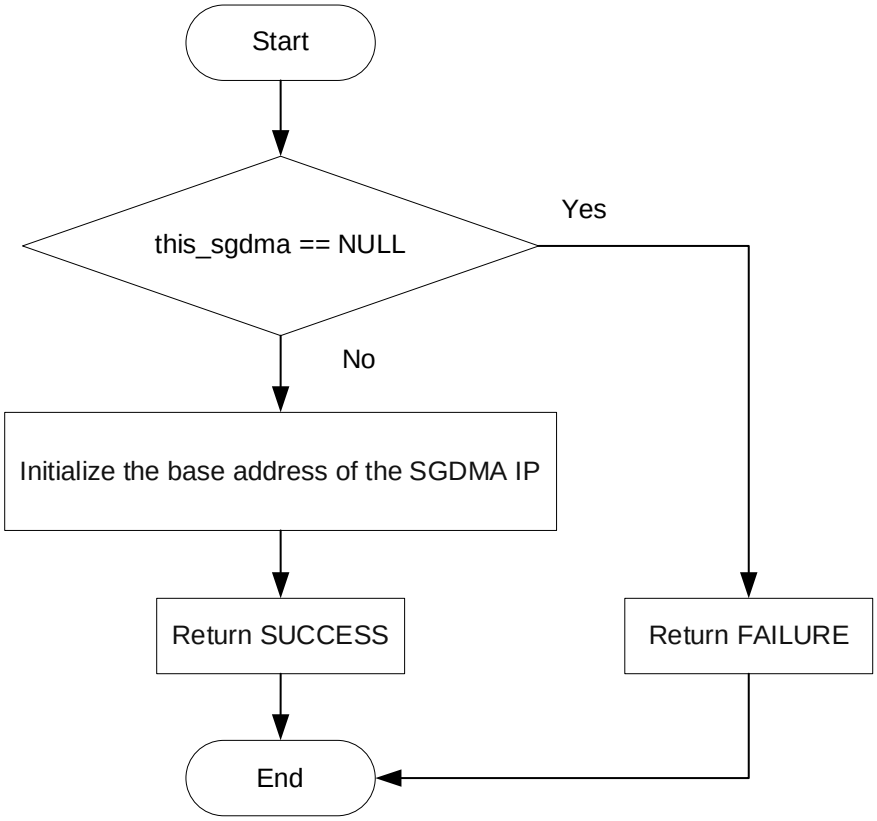

Figure 3.1. sgdma_init()

3.2. sgdma_reset_mm2s()

```
unsigned int sgdma_reset_mm2s(sgdma_instance_t *this_sgdma)
```

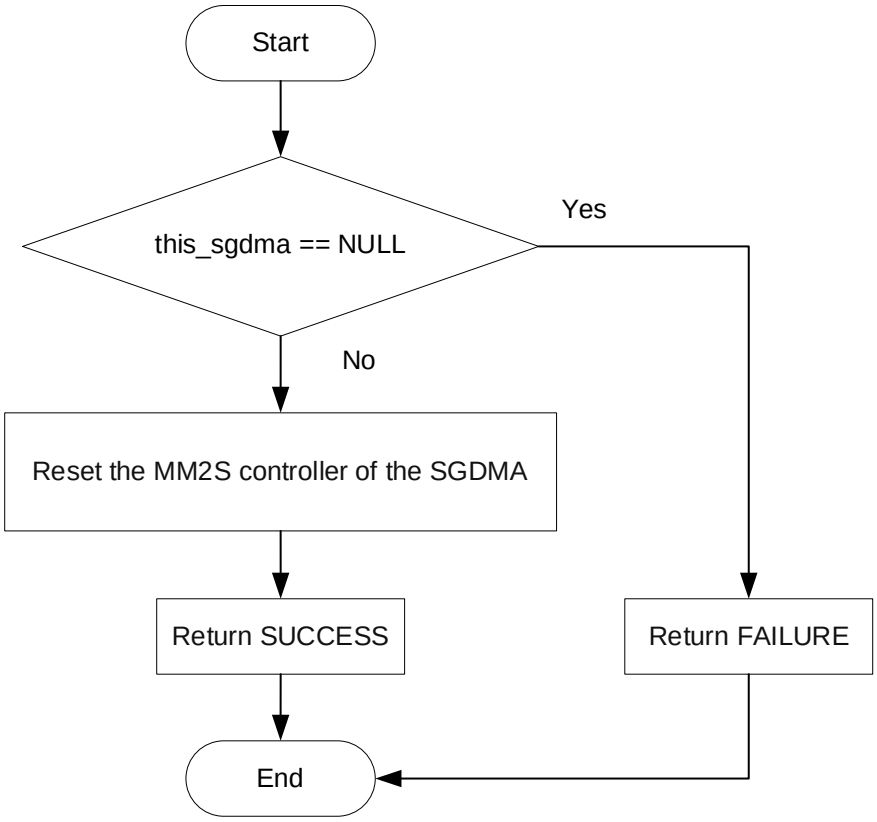

Figure 3.2. sgdma_reset_mm2s()

3.3. sgdma_reset_s2mm()

```
unsigned int sgdma_reset_s2mm(sgdma_instance_t *this_sgdma)
```

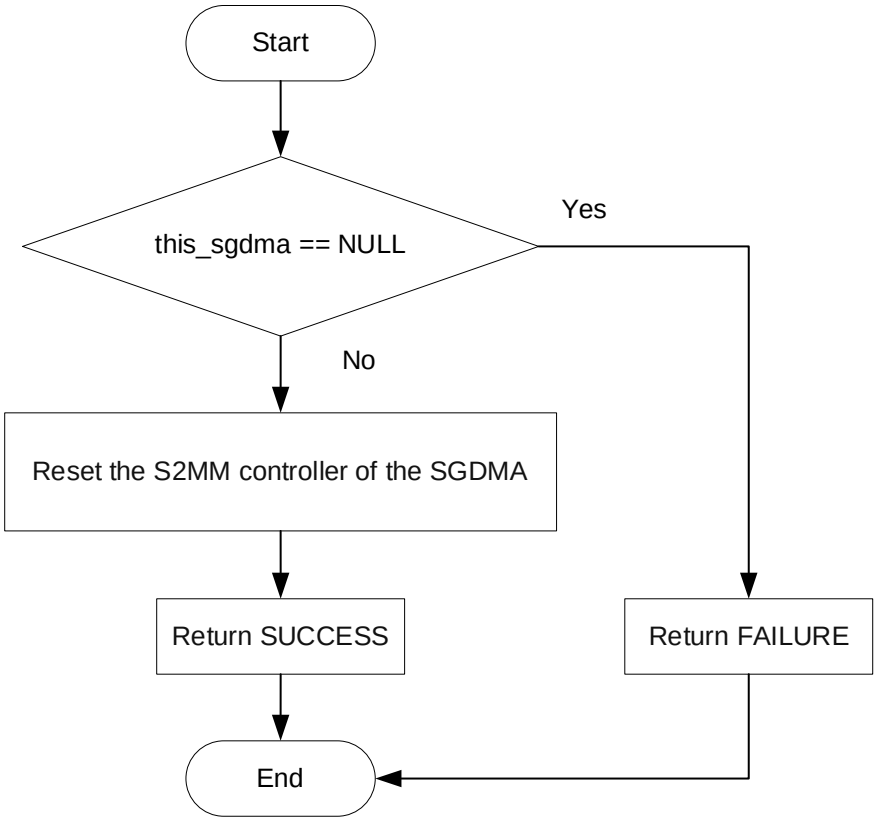

Figure 3.3. sgdma_reset_s2mm()

3.4. sgdma_reset()

```
unsigned int sgdma_reset(sgdma_instance_t *this_sgdma)
```

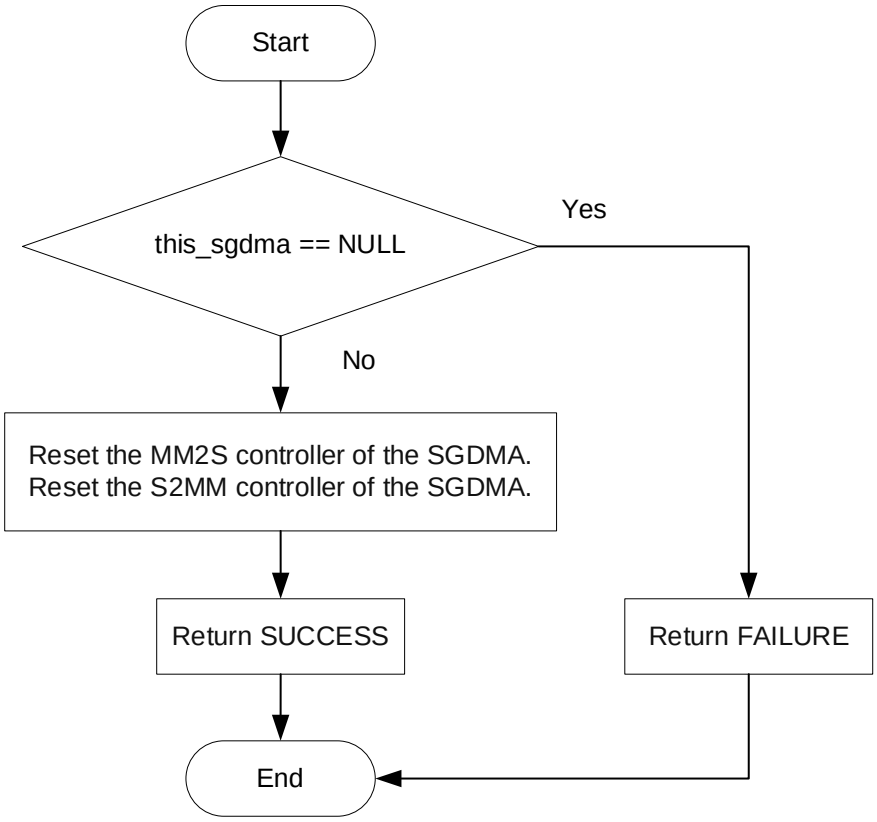

Figure 3.4. sgdma_reset()

3.5. sgdma_irq_mask()

```
unsigned int sgdma_irq_mask(sgdma_instance_t *this_sgdma, control_type_t sgdma_cntl_type,
irq_mask_t mask_value)
```

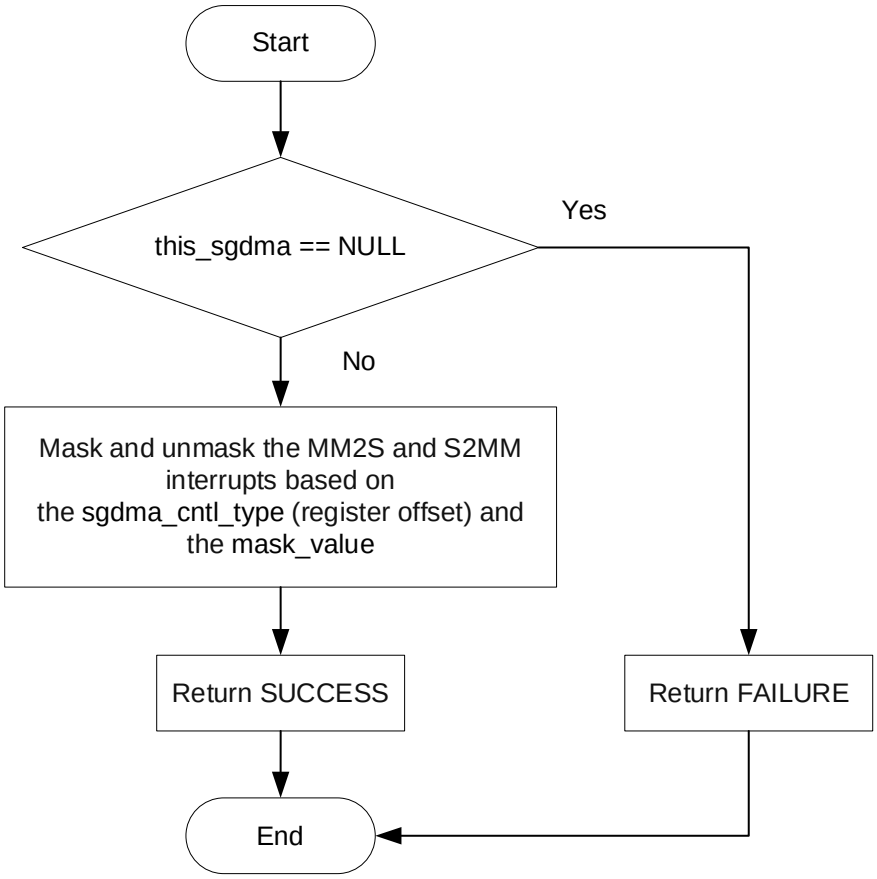

Figure 3.5. sgdma_irq_mask()

3.6. get_sgdma_s2mm_reg_status()

```
unsigned int get_sgdma_s2mm_reg_status(sgdma_instance_t *this_sgdma, sgdma_sts_reg_t *sts_reg)
```

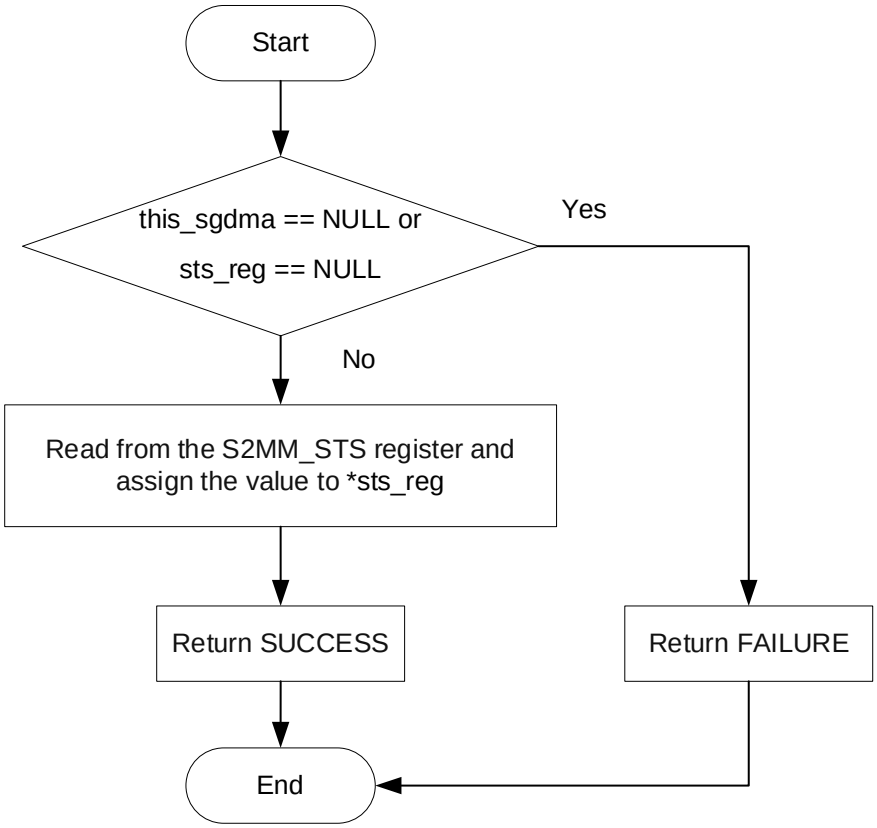

Figure 3.6. get_sgdma_s2mm_reg_status()

3.7. get_sgdma_mm2s_reg_status()

```
unsigned int get_sgdma_mm2s_reg_status(sgdma_instance_t *this_sgdma, sgdma_sts_reg_t *sts_reg)
```

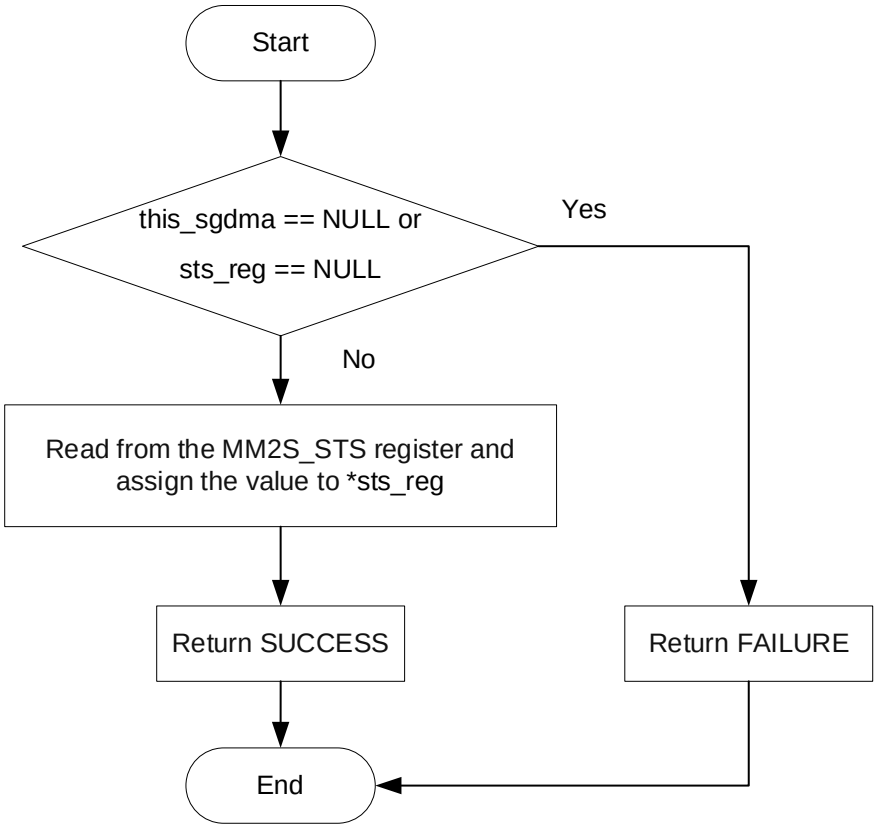

Figure 3.7. get_sgdma_mm2s_reg_status()

3.8. get_mm2s_bd_status()

```
unsigned int get_mm2s_bd_status(sgdma_instance_t *this_sgdma, int idx, unsigned int *status)
```

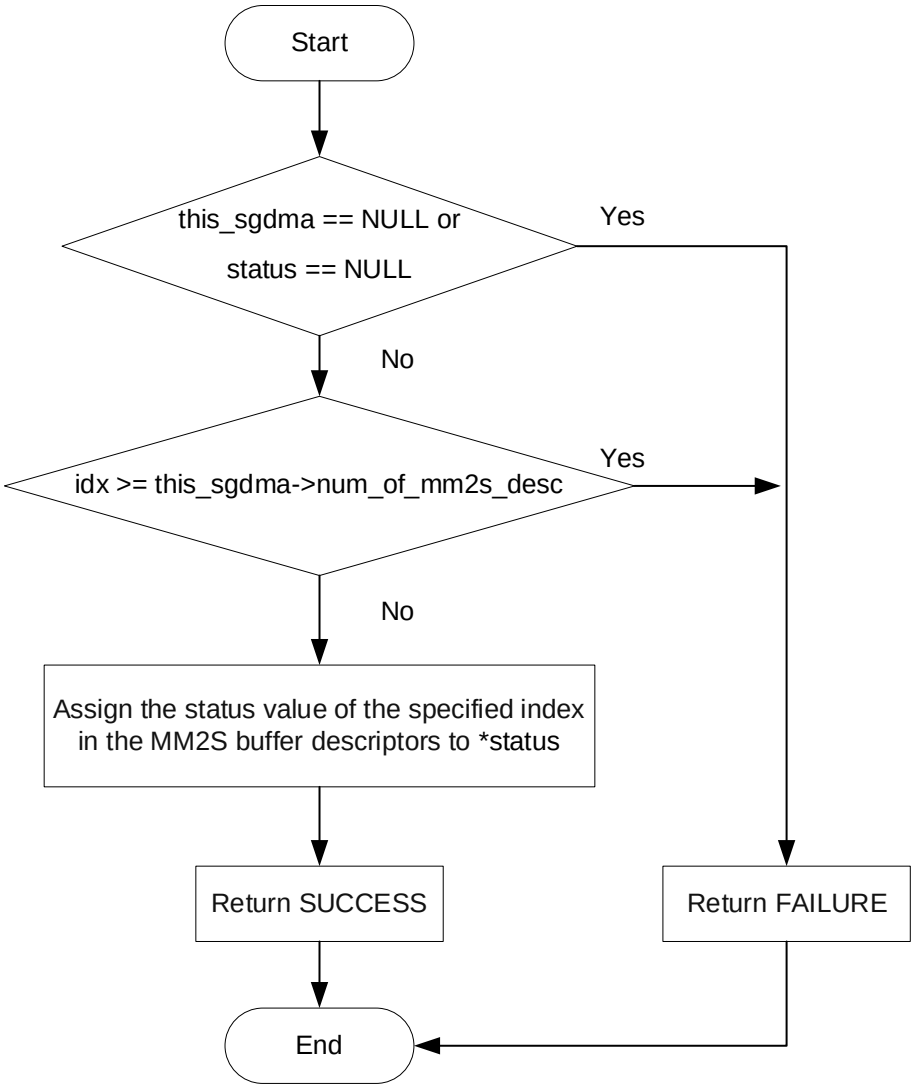

Figure 3.8. get_mm2s_bd_status()

3.9. get_s2mm_bd_status()

```
unsigned int get_s2mm_bd_status(sgdma_instance_t *this_sgdma, int idx, unsigned int *status)
```

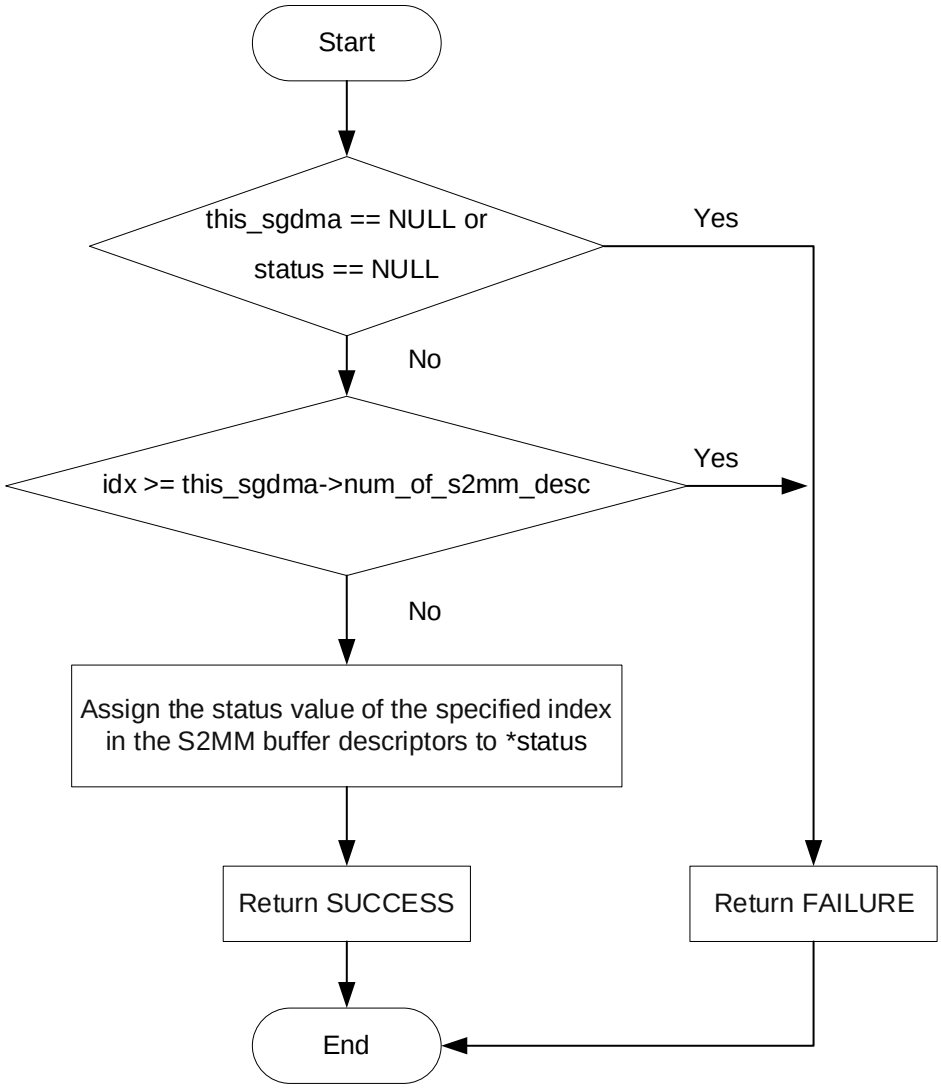

Figure 3.9. get_s2mm_bd_status()

3.10. clear_mm2s_bd_status()

```
unsigned int clear_mm2s_bd_status(sgdma_instance_t *this_sgdma, int idx)
```

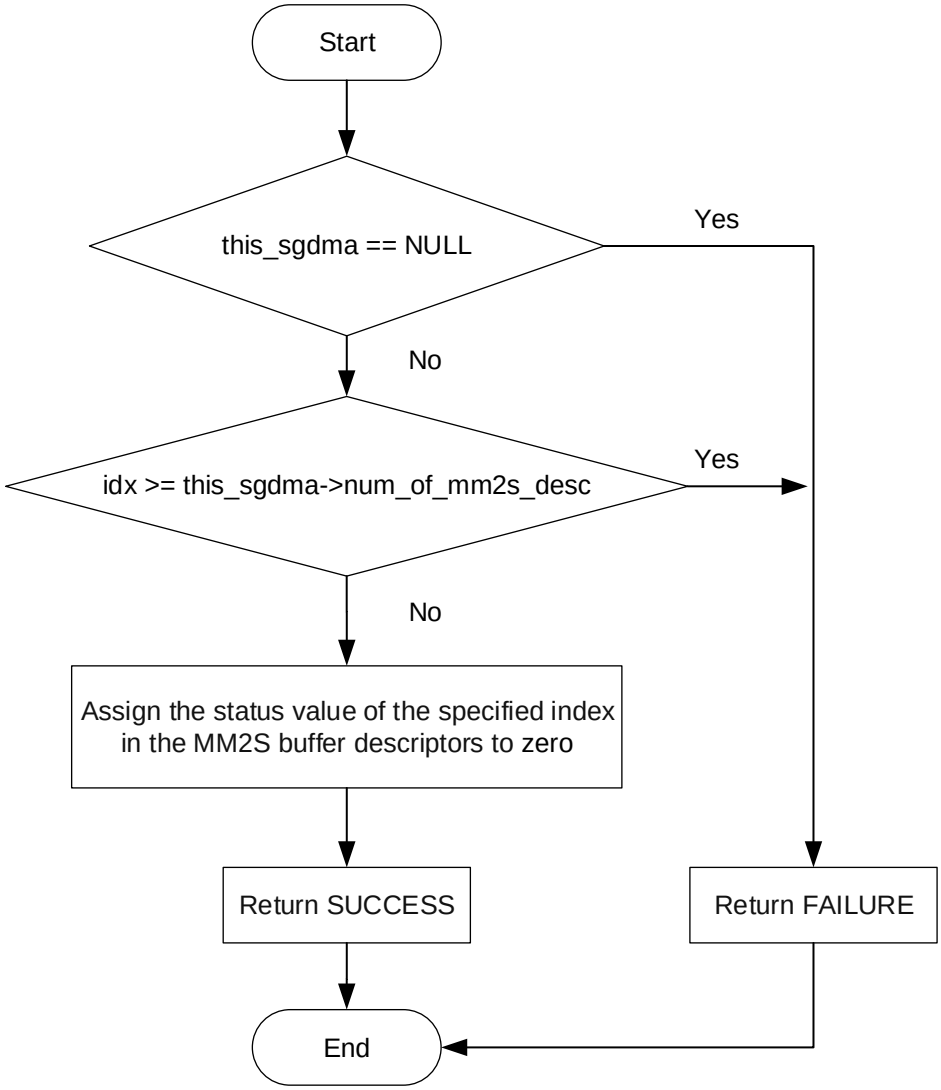

Figure 3.10. clear_mm2s_bd_status()

3.11. clear_s2mm_bd_status()

```
unsigned int clear_s2mm_bd_status(sgdma_instance_t *this_sgdma, int idx)
```

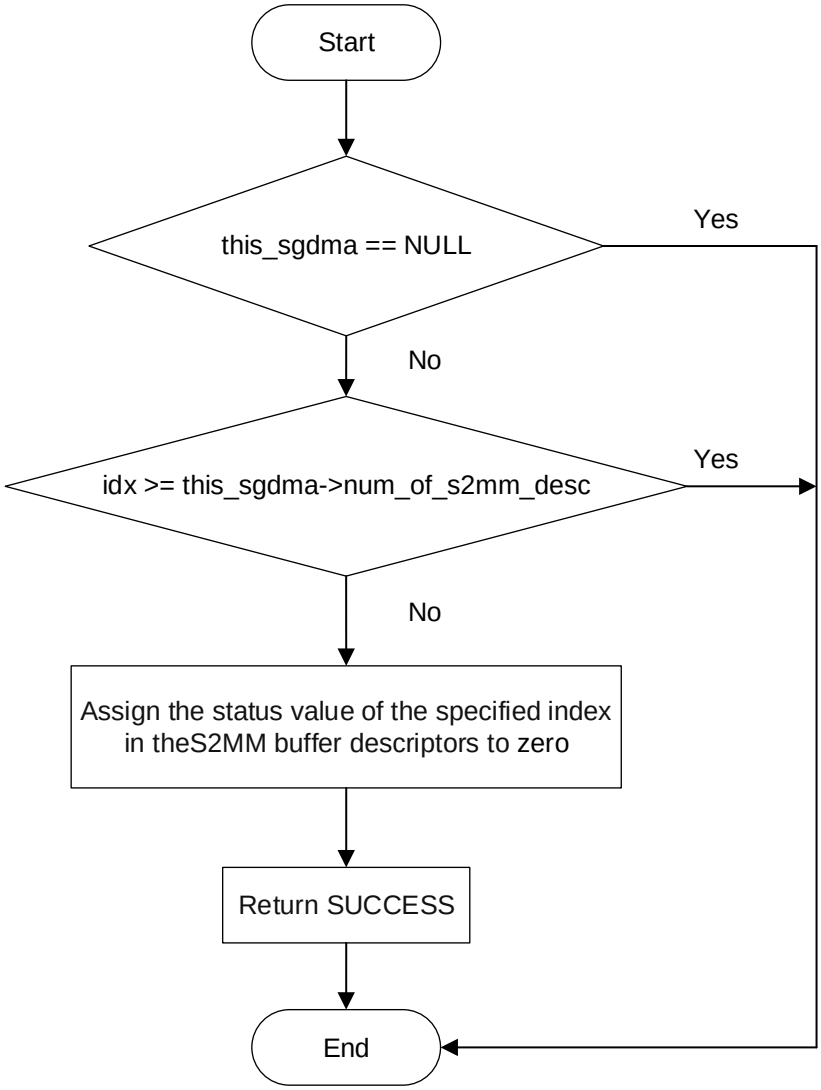

Figure 3.11. clear_s2mm_bd_status()

3.12. mm2s_start_transfer()

```
unsigned int mm2s_start_transfer(sgdma_instance_t *this_sgdma)
```

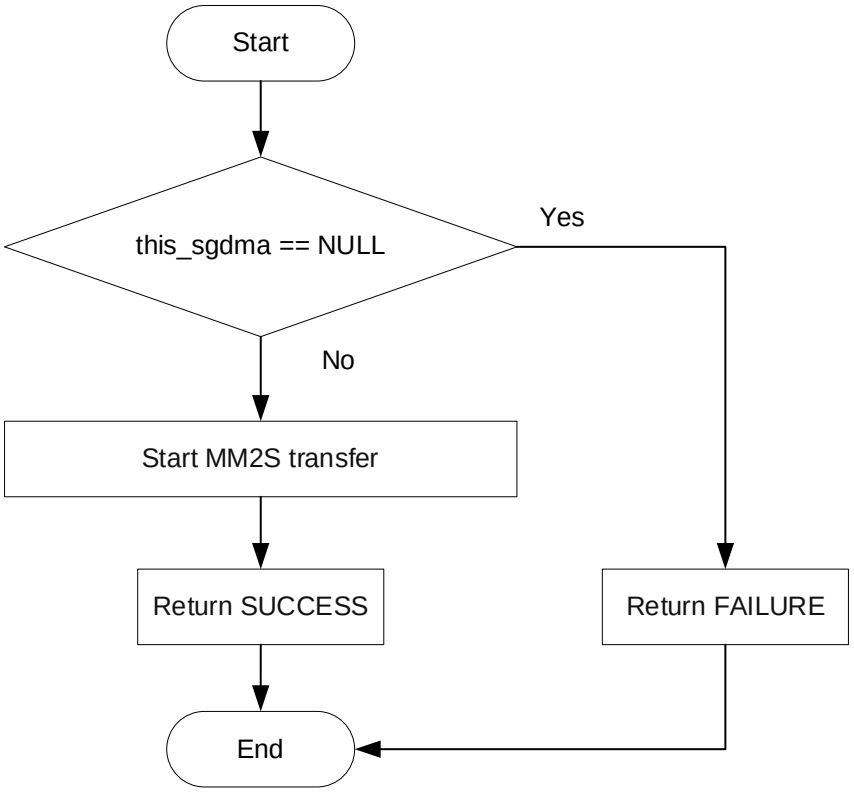

Figure 3.12. mm2s_start_transfer()

3.13. mm2s_enable_interrupt()

```
unsigned int mm2s_enable_interrupt(sgdma_instance_t *this_sgdma)
```

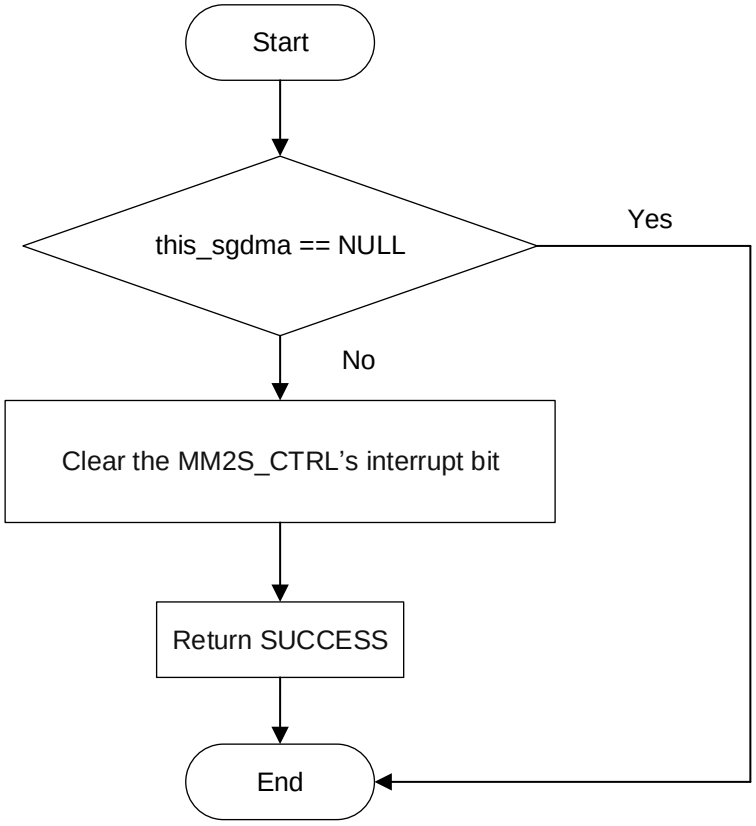

Figure 3.13. mm2s_enable_interrupt()

3.14. mm2s_disable_interrupt()

```
unsigned int mm2s_disable_interrupt(sgdma_instance_t *this_sgdma)
```

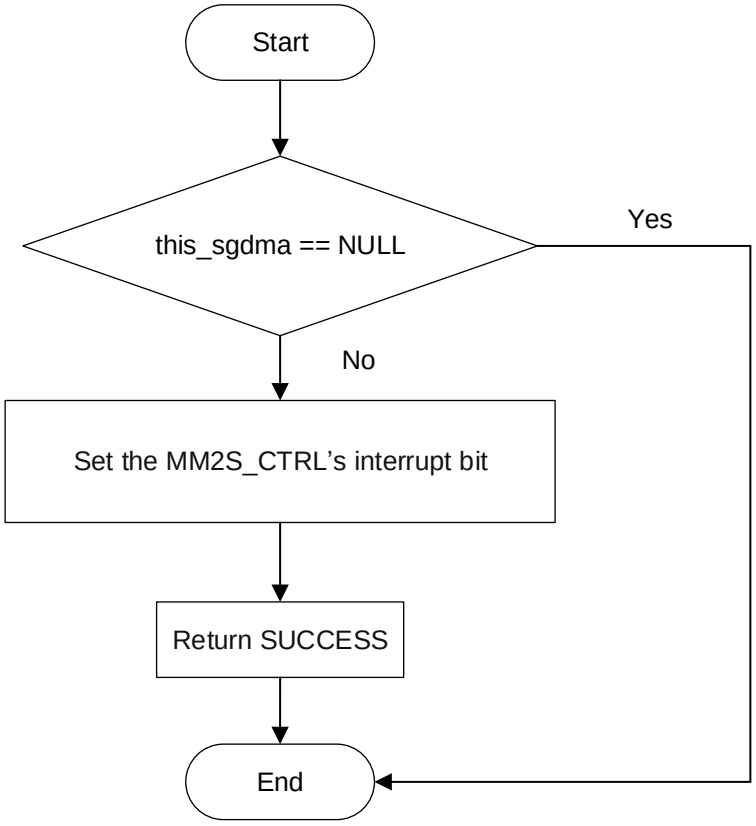

Figure 3.14. mm2s_disable_interrupt()

3.15. mm2s_get_desc_complete_bit()

```
unsigned int mm2s_get_desc_complete_bit(sgdma_instance_t *this_sgdma, unsigned int idx,
unsigned char *isComplete, unsigned char *isError)
```

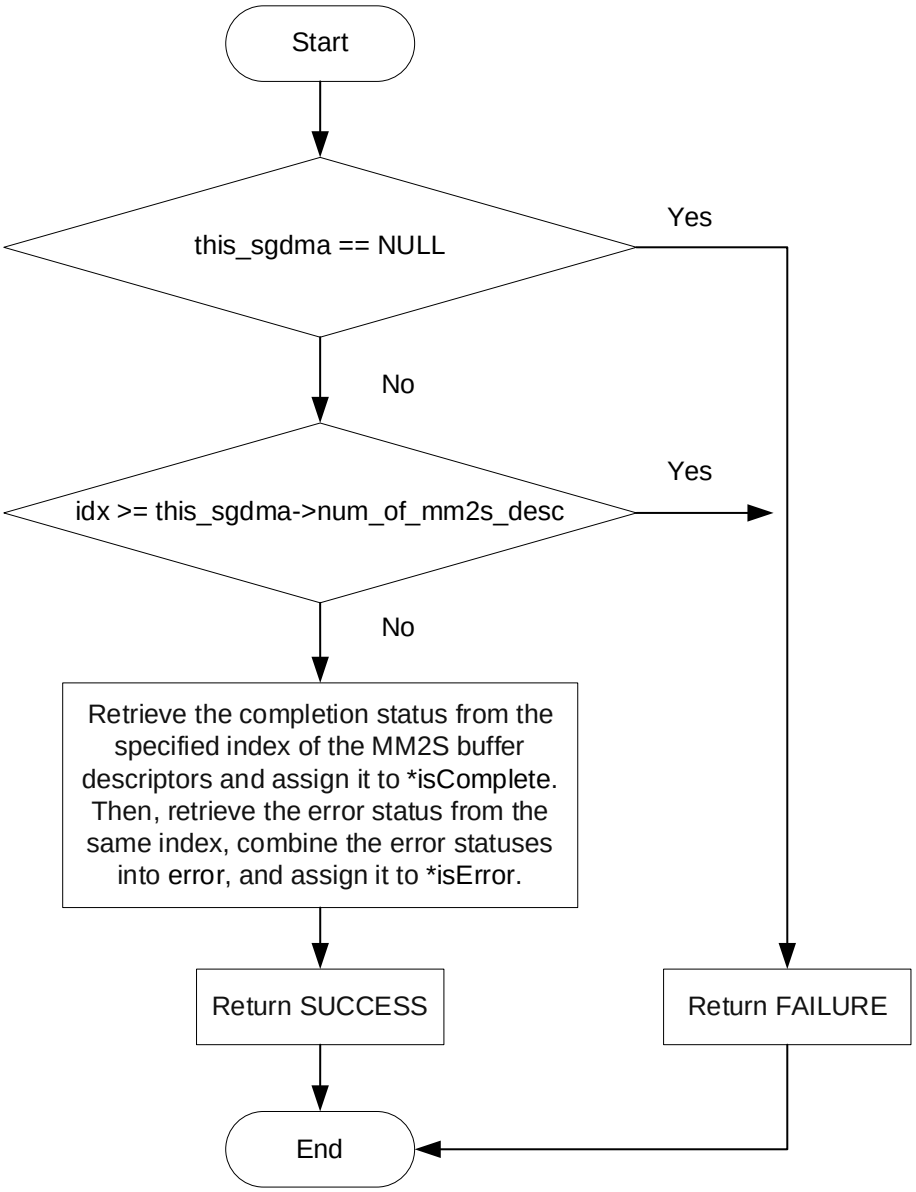

Figure 3.15. mm2s_get_desc_complete_bit()

3.16. mm2s_get_all_descs_complete_bit()

```
mm2s_get_all_descs_complete_bit(sgdma_instance_t *this_sgdma)
```

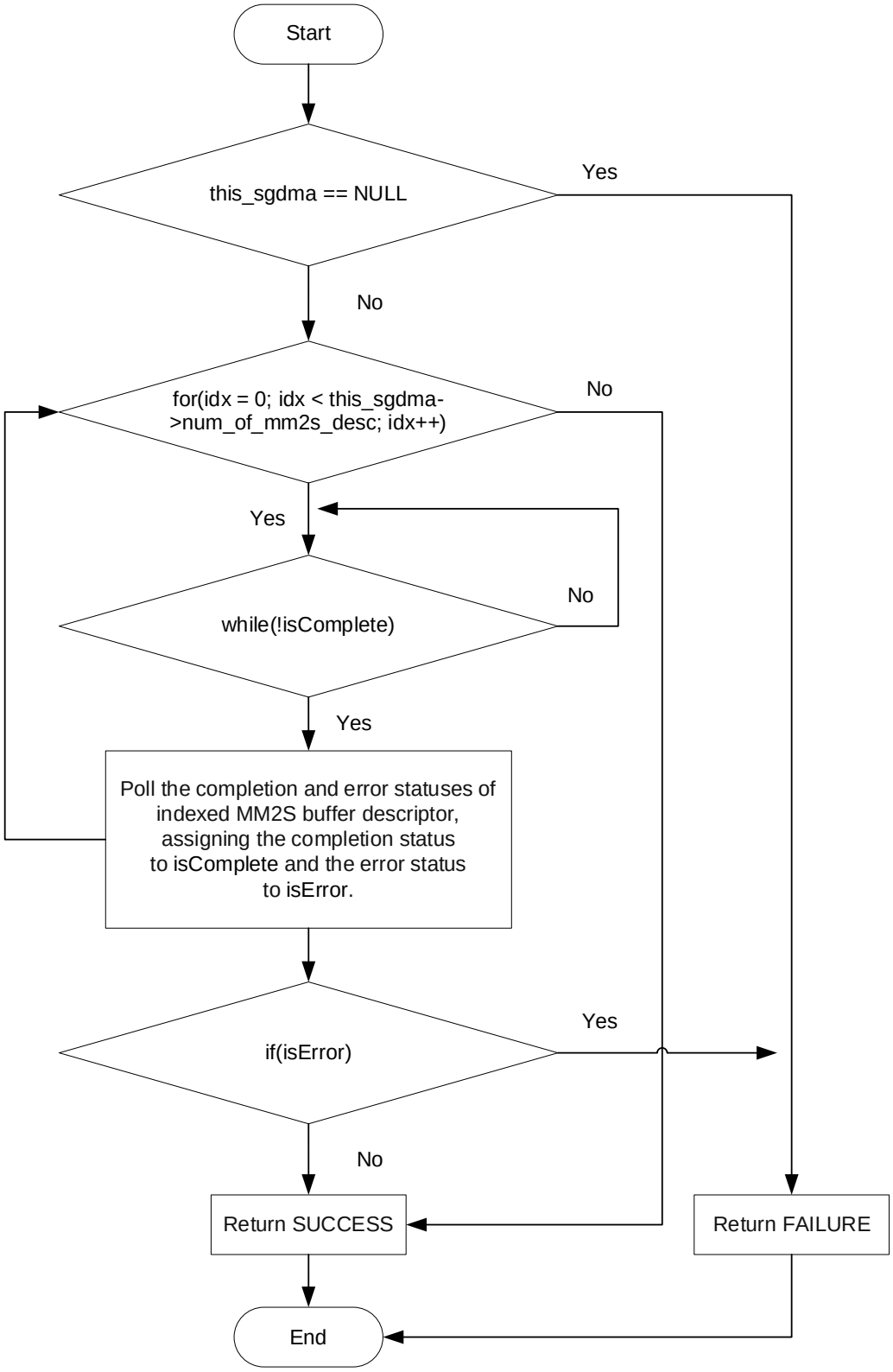

Figure 3.16. mm2s_get_all_descs_complete_bit()

3.17. mm2s_buf_desc_dma()

```
unsigned int mm2s_buf_desc_dma(sgdma_instance_t *this_sgdma)
```

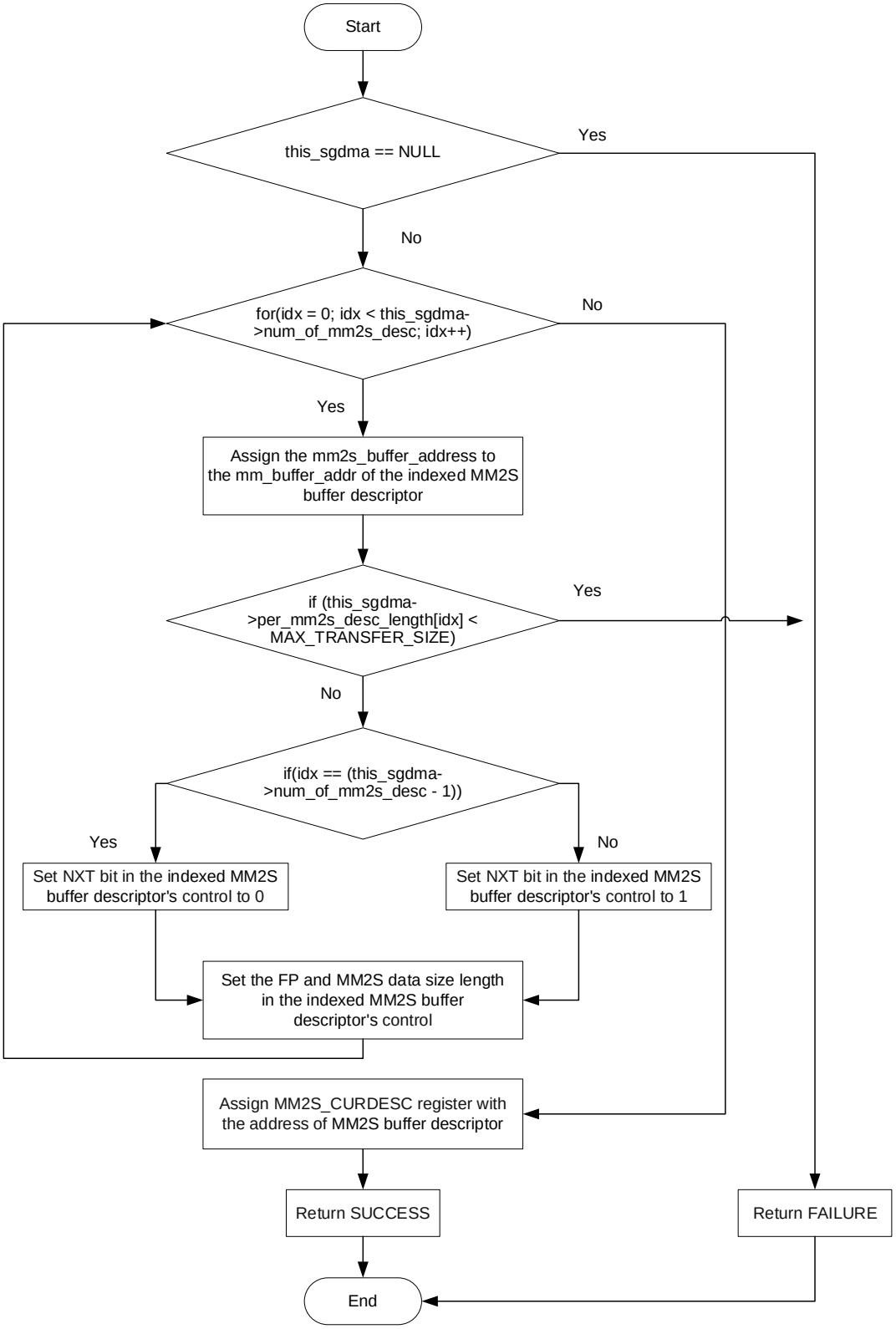

Figure 3.17. mm2s_buf_desc_dma()

3.18. s2mm_start_transfer()

```
unsigned int s2mm_start_transfer(sgdma_instance_t *this_sgdma)
```

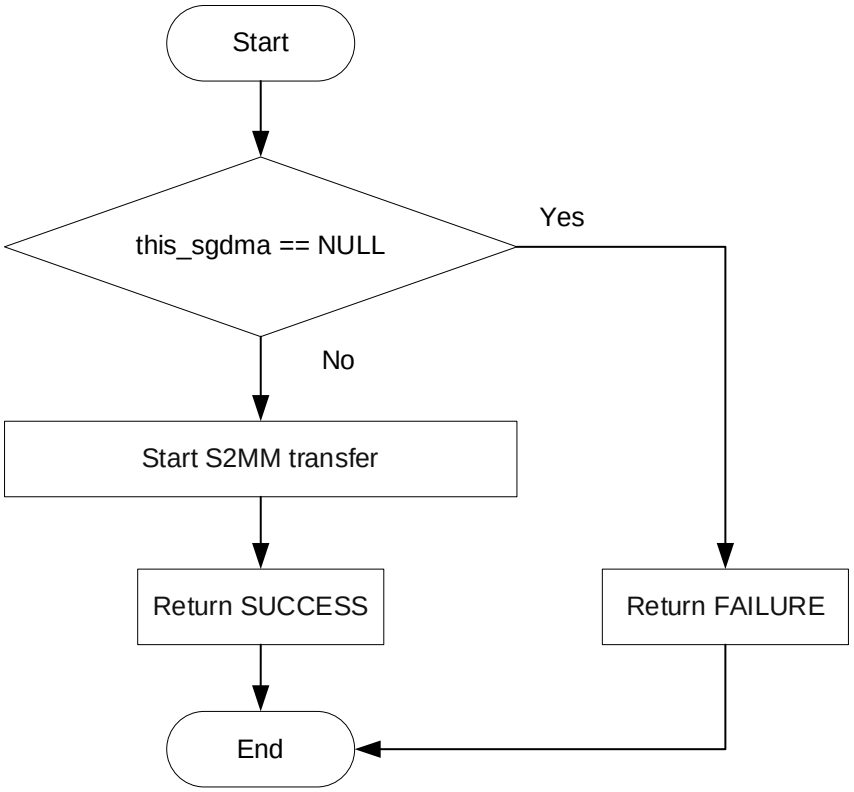

Figure 3.18. s2mm_start_transfer()

3.19. s2mm_enable_interrupt()

```
unsigned int s2mm_enable_interrupt(sgdma_instance_t *this_sgdma)
```

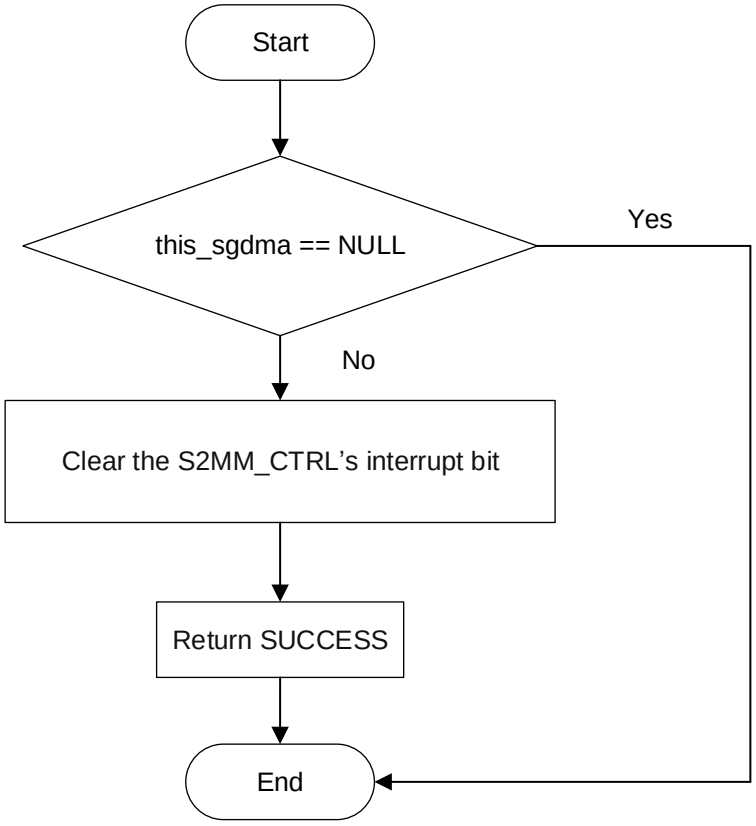

Figure 3.19. s2mm_enable_interrupt()

3.20. s2mm_disable_interrupt()

```
unsigned int s2mm_disable_interrupt(sgdma_instance_t *this_sgdma)
```

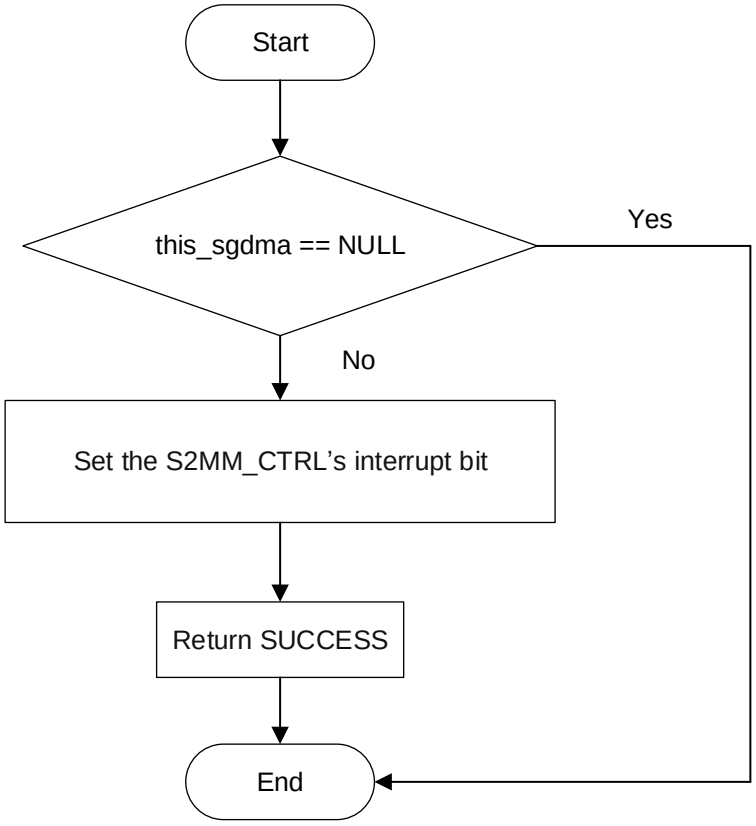

Figure 3.20. s2mm_disable_interrupt()

3.21. s2mm_get_desc_complete_bit()

```
unsigned int s2mm_get_desc_complete_bit(sgdma_instance_t *this_sgdma, unsigned int idx,
unsigned char *isComplete, unsigned char *isError)
```

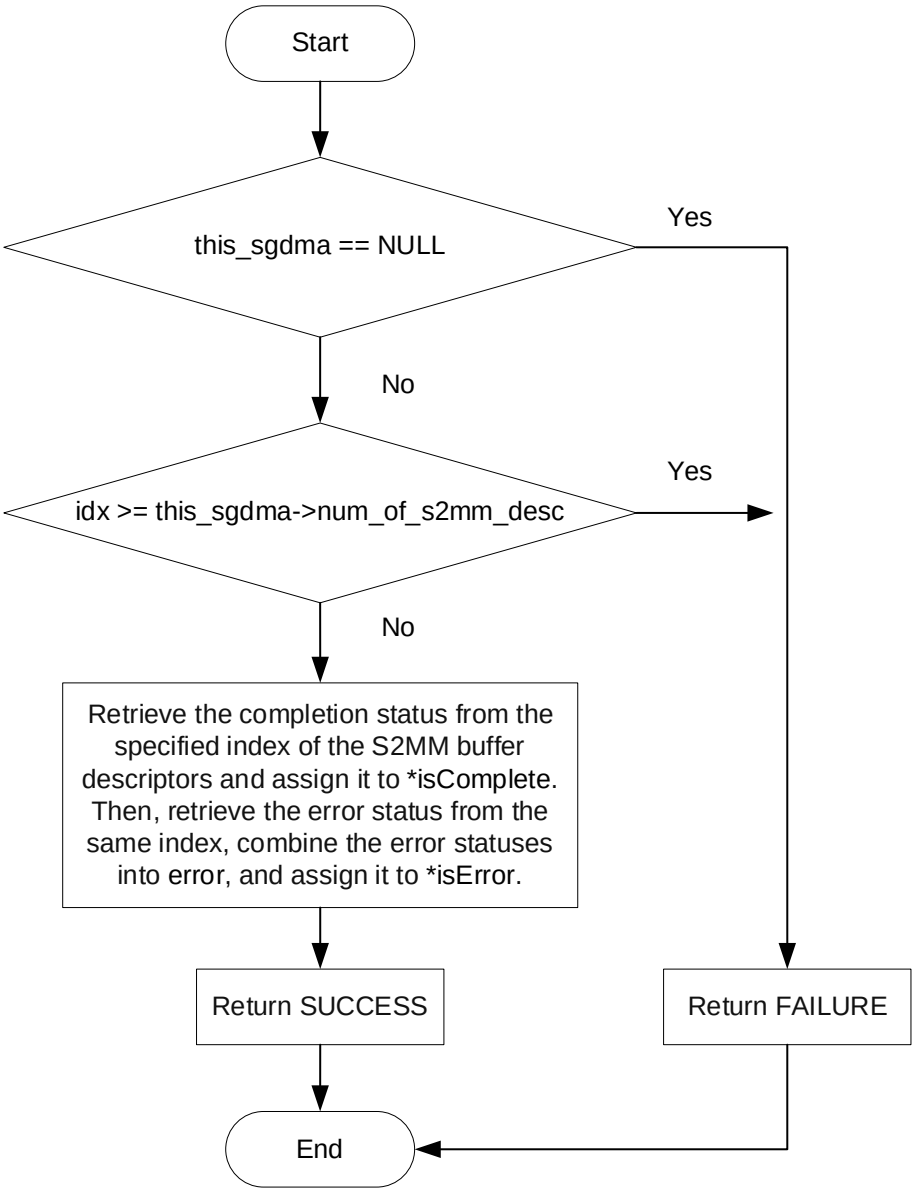

Figure 3.21. s2mm_get_desc_complete_bit()

3.22. s2mm_get_all_descs_complete_bit()

```
unsigned int s2mm_get_all_desc_complete_bit(sgdma_instance_t *this_sgdma)
```

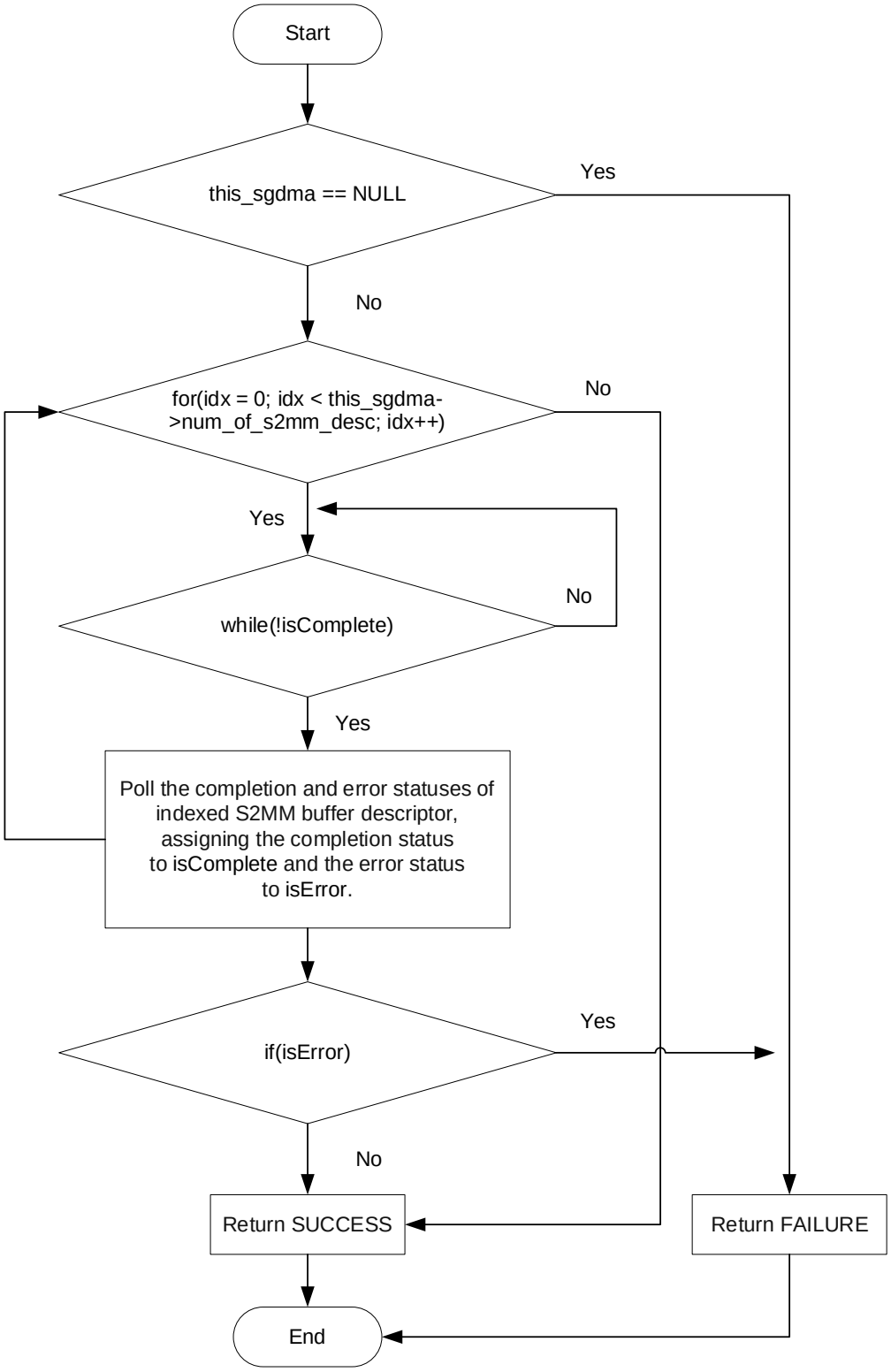

Figure 3.22. s2mm_get_all_descs_complete_bit()

3.23. s2mm_buf_desc_dma()

```
unsigned int s2mm_buf_desc_dma(sgdma_instance_t *this_sgdma)
```

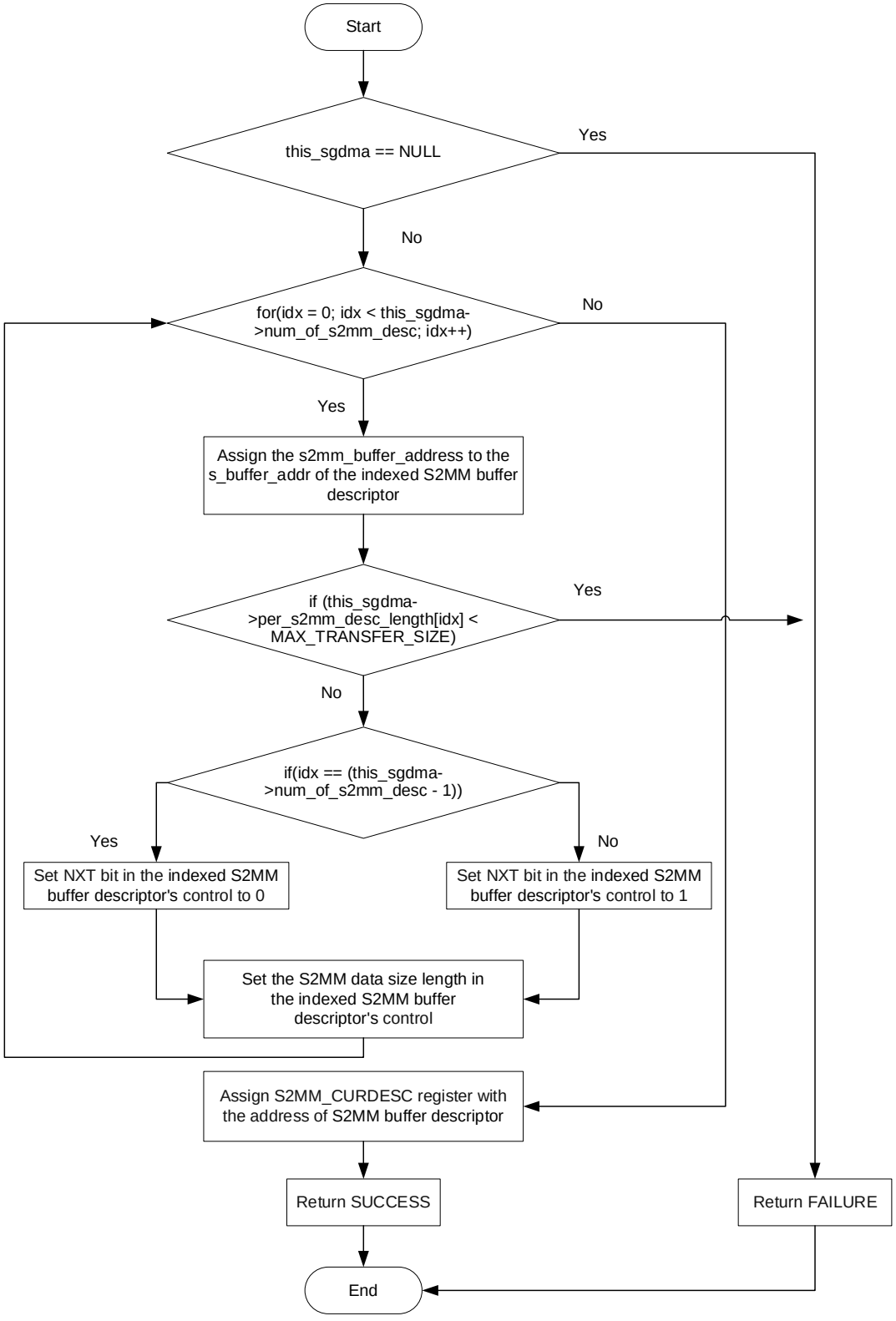

Figure 3.23. s2mm_buf_desc_dma()

3.24. sgdma_register_read()

```
unsigned int sgdma_register_read(sgdma_instance_t *this_sgdma, sgdma_reg_type_t index, unsigned int *reg_data)
```

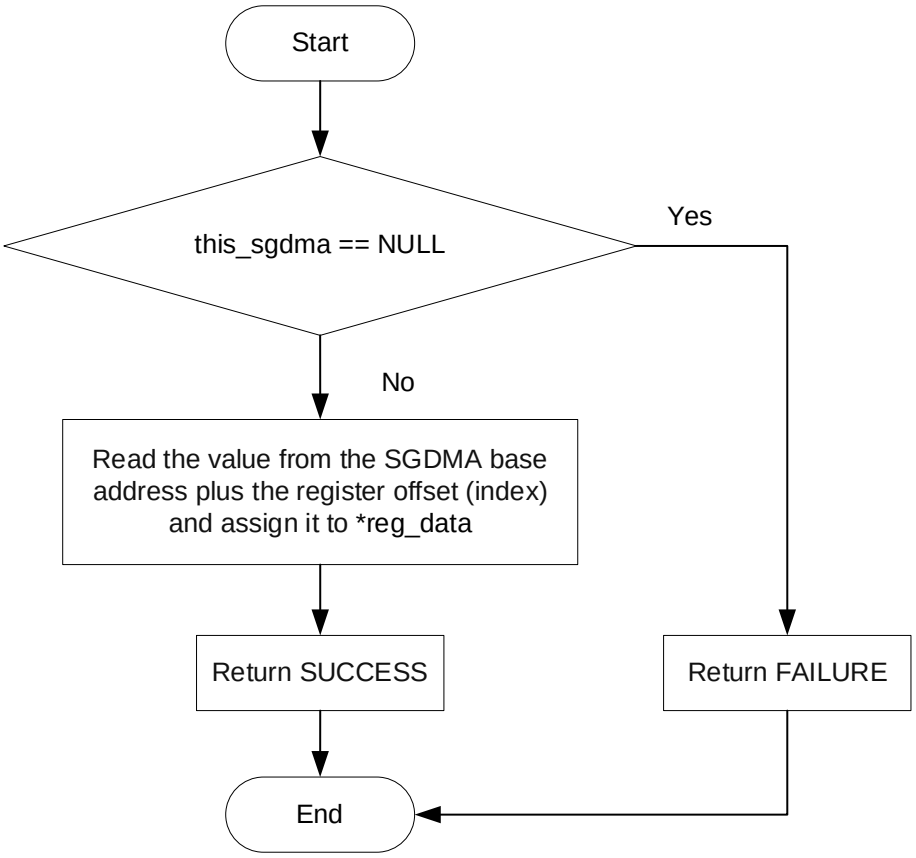

Figure 3.24. sgdma_register_read()

3.25. sgdma_register_write()

```
unsigned int sgdma_register_write(sgdma_instance_t *this_sgdma, sgdma_reg_type_t index,
unsigned int reg_data)
```

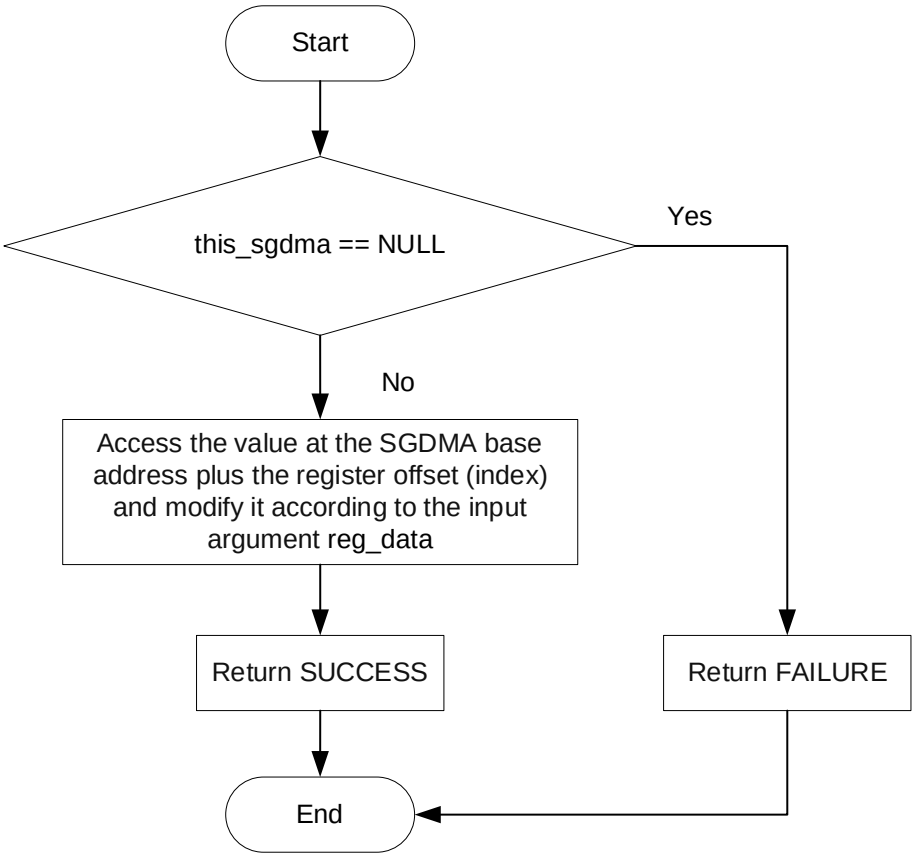

Figure 3.25. sgdma_register_write()

4. API Data Structure and Enums

4.1. struct mm2s_ctrl_reg_t

Table 4.1. struct mm2s_ctrl_reg_t

Data Type	Structure member	Bit/Bitfield
volatile unsigned int	mm_request	1
volatile unsigned int	mm_reset	1
volatile unsigned int	mm_rsvd_1	14
volatile unsigned int	mm_cmpl_irq_mask	1
volatile unsigned int	mm_rsvd_2	15

4.2. struct mm2s_sts_reg_t

Table 4.2. struct mm2s_sts_reg_t

Data Type	Structure member	Bit/Bitfield
volatile unsigned int	mm_status	1
volatile unsigned int	mm_rsvd_1	7
volatile unsigned int	mm_reg_bd_len_err	1
volatile unsigned int	mm_reg_axi_slave_err	1
volatile unsigned int	mm_reg_axi_desc_err	1
volatile unsigned int	mm_rsvd_2	5
volatile unsigned int	mm_xfer_cmpl	1
volatile unsigned int	mm_xfer_err	1
volatile unsigned int	mm_rsvd_3	14

4.3. struct mm2s_desc_tx_t

Table 4.3. struct mm2s_desc_tx_t

Data Type	Structure member	Bit/Bitfield
volatile unsigned int	mm_buffer_addr	32
volatile unsigned int	mm_buffer_msb_addr	32
volatile unsigned int	mm_control_buffer_size	16
volatile unsigned int	mm_control_tdest	4
volatile unsigned int	mm_control_tid	4
volatile unsigned int	mm_control_arprot	3
volatile unsigned int	mm_control_rsvd	3
volatile unsigned int	mm_control_fp	1
volatile unsigned int	mm_control_nxt	1
volatile unsigned int	mm_status_transffered_size	16
volatile unsigned int	mm_status_rsvd	11
volatile unsigned int	mm_status_axi_desc_err	1
volatile unsigned int	mm_status_axi_slave_err	1
volatile unsigned int	mm_status_transffered_len_err	1
volatile unsigned int	mm_status_bd_len_err	1
volatile unsigned int	mm_status_cmpl	1

4.4. struct mm2s_desc_tx_ext_t

Table 4.4. struct mm2s_desc_tx_ext_t

Data Type	Structure member
volatile unsigned int	mm_buffer_addr
volatile unsigned int	mm_buffer_msb_addr
volatile unsigned int	mm_control
volatile unsigned int	mm_status

4.5. union mm2s_desc_t

Table 4.5. union mm2s_desc_t

Data Type	Union member
mm2s_desc_tx_ext_t	mm_bd_ext
mm2s_desc_tx_t	mm_bd

4.6. struct s2mm_ctrl_reg_t

Table 4.6. struct s2mm_ctrl_reg_t

Data Type	Structure member	Bit/Bitfield
volatile unsigned int	s_request	1
volatile unsigned int	s_reset	1
volatile unsigned int	s_rsvd_1	14
volatile unsigned int	s_cmpl_irq_mask	1
volatile unsigned int	s_rsvd_2	15

4.7. struct s2mm_sts_reg_t

Table 4.7. struct s2mm_sts_reg_t

Data Type	Structure member	Bit/Bitfield
volatile unsigned int	s_status	1
volatile unsigned int	s_rsvd_1	7
volatile unsigned int	s_reg_bd_len_err	1
volatile unsigned int	s_reg_axi_slave_err	1
volatile unsigned int	s_reg_axi_desc_err	1
volatile unsigned int	s_rsvd_2	5
volatile unsigned int	s_xfer_cmpl	1
volatile unsigned int	s_xfer_err	1
volatile unsigned int	s_rsvd_3	14

4.8. struct s2mm_desc_tx_t

Table 4.8. struct s2mm_desc_tx_t

Data Type	Structure member	Bit/Bitfield
volatile unsigned int	s_buffer_addr	32
volatile unsigned int	s_buffer_msb_addr	32
volatile unsigned int	s_control_buffer_size	16
volatile unsigned int	s_control_rsvd_1	8
volatile unsigned int	s_control_awprot	3
volatile unsigned int	s_control_rsvd_2	4
volatile unsigned int	s_control_nxt	1
volatile unsigned int	s_status_transffered_size	16
volatile unsigned int	s_status_rsvd	11
volatile unsigned int	s_status_axi_desc_err	1
volatile unsigned int	s_status_axi_slave_err	1
volatile unsigned int	s_status_transffered_len_err	1
volatile unsigned int	s_status_bd_len_err	1
volatile unsigned int	s_status_cmpl	1

4.9. struct s2mm_desc_tx_ext_t

Table 4.9. struct s2mm_desc_tx_ext_t

Data Type	Structure member
volatile unsigned int	s_buffer_addr
volatile unsigned int	s_buffer_msb_addr
volatile unsigned int	s_control
volatile unsigned int	s_status

4.10. union s2mm_desc_t

Table 4.10. union s2mm_desc_t

Data Type	Union member
s2mm_desc_tx_ext_t	s_bd_ext
s2mm_desc_tx_t	s_bd

4.11. struct sgdma_instance_t

Table 4.11. struct sgdma_instance_t

Data Type	Structure member
unsigned int	base_addr
unsigned int	*mm2s_buffer_addr
unsigned int	*s2mm_buffer_addr
unsigned int	axi4_mm_data_width
unsigned int	num_of_mm2s_desc
unsigned int	num_of_s2mm_desc
mm2s_desc_t	*mm2s_bd_addr
s2mm_desc_t	*s2mm_bd_addr
unsigned int	*per_mm2s_desc_length
unsigned int	*per_s2mm_desc_length
unsigned int	fp

4.12. struct sgdma_ctrl_reg_t

Table 4.12. struct sgdma_ctrl_reg_t

Data Type	Structure member	Bit/Bitfield
volatile unsigned int	request	1
volatile unsigned int	reset	1
volatile unsigned int	rsvd_1	14
volatile unsigned int	cmpl_irq_mask	1
volatile unsigned int	rsvd_2	15

4.13. struct sgdma_sts_reg_t

Table 4.13. struct sgdma_sts_reg_t

Data Type	Structure member	Bit/Bitfield
volatile unsigned int	status	1
volatile unsigned int	rsvd_1	7
volatile unsigned int	reg_bd_len_err	1
volatile unsigned int	reg_axi_slave_err	1
volatile unsigned int	reg_axi_desc_err	1
volatile unsigned int	rsvd_2	5
volatile unsigned int	xfer_cmpl	1
volatile unsigned int	xfer_err	1
volatile unsigned int	rsvd_3	14

4.14. enum control_type_t

Table 4.14. enum control_type_t

Data Type	Value
MM2S_CONTROL	0
S2MM_CONTROL	0x20

4.15. enum status_type_t

Table 4.15. enum status_type_t

Data Type	Value
MM2S_STATUS	0x4
S2MM_STATUS	0x24

4.16. enum irq_mask_t

Table 4.16. enum irq_mask_t

Data Type	Value
UNMASK	0
MASK	1

4.17. enum sgdma_reg_type_t

Table 4.17. enum sgdma_reg_type_t

Data Type	Value
MM2S_CTRL_REG	0x0
MM2S_STATUS_REG	0x4
MM2S_CURDESC_REG	0x8
S2MM_CTRL_REG	0x20
S2MM_STATUS_REG	0x24
S2MM_CURDESC_REG	0x28

5. API Macros

Table 5.1. API Macros

Macro name	Description
MM2S_CTRL	mm2s control offset
MM2S_STS	mm2s status offset
MM2S_CURDESC	mm2s current descriptor offset
S2MM_CTRL	s2mm control offset
S2MM_STS	s2mm status offset
S2MM_CURDESC	s2mm current descriptor offset
MAX_TRANSFER_SIZE	max total data bytes data transferred
MULTI_BUF_SIZE	multiple buffer descriptor number
FP_BIT_POSITION	full packet bit position
NXT_BIT_POSITION	next bit position
TID_BIT_POSITION	tid bit position
TDEST_BIT_POSITION	tdest bit position
XFER_CMPL_BIT_POSITION	xfer_cmpl bit position
DESC_SIZE	size of a buffer descriptor
BUFFER_SIZE_MASK	use masking for extract buffer size
SGDMA_RESET	reset control register of s2mm and mm2s
DMA_TRIGGER	dma trigger value
MSB_ADDR	use for msb address
NXT_1	1 for multiple descriptor
NXT_0	0 for last descriptor
FP_1	full packet pre-fetch required
FP_0	full packet pre-fetch not required
XFER_CMPL_1	use for bit operation
IDX_0	use for index zero
TRUE	1 for conditional operation
FALSE	0 for conditional operation
FAILURE	0
SUCCESS	1
IS_NULL	0

References

- SGDMA Controller IP Core - Lattice Radiant Software User Guide (FPGA-IPUG- 02131)
- SDGMA Controller IP Release Notes (FPGA-RN-02058)
- Avant-E web page
- Avant-G web page
- Avant-X web page
- CertusPro-NX web page
- Lattice Solutions IP Cores web page
- Lattice Radiant Software FPGA web page
- Lattice Insights web page 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
Introduction	• Updated the paragraph including adding reference to the SGDMA IPUG in the Audience section. • Updated the SGDMA version in the Driver Version section. • Updated Table 1.1. Driver and IP Compatibility and Table 1.2. Quick Facts of Driver Tested Environment.

API Description	• Updated the programming codes from void to *unsigned int* in the following sections:
	  • sgdma_init()
	  • sgdma_reset_mm2s()
	  • sgdma_reset_s2mm()
	  • sgdma_reset()
	  • sgdma_irq_mask()
	  • mm2s_buf_desc_dma()
	  • s2mm_buf_desc_dma()
	  • sgdma_register_read()
	  • sgdma_register_write()
	• Updated the start of programming code from void get_sgdma_ reg_status to *unsigned int get_sgdma_mm2s_reg_status* table name to Table 2.7. get_sgdma_mm2s_reg_status() in get_sgdma_mm2s_reg_status().
	• Added the following sections:
	  • clear_mm2s_bd_status()
	  • clear_s2mm_bd_status()
	  • mm2s_enable_interrupt()
	  • mm2s_disable_interrupt()
	  • s2mm_enable_interrupt()
	  • s2mm_disable_interrupt()

Functional Call Flow Diagrams	• Added the following sections:
	  • clear_mm2s_bd_status()
	  • clear_s2mm_bd_status()
	  • mm2s_enable_interrupt()
	  • mm2s_disable_interrupt()
	  • s2mm_enable_interrupt()
	  • s2mm_disable_interrupt()

Revision 1.3, February 2025

Section	Change Summary
All	• Removed the *API Variables* section. • Made editorial fixes.
Introduction	Updated the following sections: • Audience • Driver Version • Driver and IP Compatibility

API Description	• Added an introductory paragraph in the API Description section.
	• Updated the descriptions and tables in the following sections:
	  • sgdma_init()
	  • sgdma_reset()
	  • sgdma_irq_mask()
	  • get_sgdma_mm2s_reg_status()
	  • get_mm2s_bd_status()

Section	Change Summary
	• get_s2mm_bd_status() • mm2s_buf_desc_dma() • s2mm_buf_desc_dma() • sgdma_register_read() • sgdma_register_write() • Updated the *this_sgdma descriptions in Table 2.11. mm2s_get_desc_complete_bit() and Table 2.15. s2mm_get_desc_complete_bit(). • Added the following sections: • sgdma_reset_mm2s() • sgdma_reset_s2mm() • get_sgdma_s2mm_reg_status() • mm2s_start_transfer() • mm2s_get_all_descs_complete_bit() • s2mm_start_transfer() • s2mm_get_all_descs_complete_bit()
Function Call Flow Diagrams	• Removed the *get_sgdma_reg_status()* section. • Added the following sections: • sgdma_reset_mm2s() • sgdma_reset_s2mm() • get_sgdma_s2mm_reg_status() • get_sgdma_mm2s_reg_status() • mm2s_start_transfer() • mm2s_get_all_descs_complete_bit() • s2mm_start_transfer() • s2mm_get_all_descs_complete_bit() • Updated the programming codes and figures in the following sections: • sgdma_init() • sgdma_reset() • sgdma_irq_mask() • get_mm2s_bd_status() • get_s2mm_bd_status() • mm2s_get_desc_complete_bit() • mm2s_buf_desc_dma() • s2mm_get_desc_complete_bit() • s2mm_buf_desc_dma() • sgdma_register_read() • sgdma_register_write()
API Data Structure and Enums	• Removed the blocking_S2MM and blocking_MM2S members from Table 4.1. struct mm2s_ctrl_reg_t. • Updated the data types in Table 4.14. enum control_type_t.
API Macros	• In Table 5.1. API Macros: • Removed the *RET_FAILURE* and *RET_SUCCESS* macros. • Added the *FAILURE* and *SUCCESS* macros.
References	Added the *Avant-E*, *Avant-G*, and *Avant-X* web pages and the *SDGMA Controller IP Release Notes (FPGA-RN-02058)* document.

Revision 1.2, July 2024

Section	Change Summary
Introduction	• Updated SGDMA version from 1.0.0 to *24.01.00*. • Updated Driver and IP versions in Table 1.1. Driver and IP Compatibility. • Added Table 1.2. Quick Facts of Driver Tested Environment.
API Description	• Updated sgdma_init() to remove *unsigned int num_of_desc* in the programming code. • Updated sgdma_reset() to remove *sgdma_cntl_type* in the programming code. • Updated API description of get_sgdma_reg_status() section. • Added mm2s_get_desc_complete_bit() and s2mm_get_desc_complete_bit(). • Updated tables in mm2s_buf_desc_dma() and s2mm_buf_desc_dma() to change Returns column value to *1 – success* and *0 – failure*. • Updated API descriptions and tables of sgdma_register_read() and sgdma_register_write() to change Returns column value to *1 – success* and *0 – failure*.
Function Call Flow Diagrams	• Added programming codes in sgdma_init() to sgdma_register_write(). • Added mm2s_get_desc_complete_bit and s2mm_get_desc_complete_bit sections.

API Data Structure and Enums	• Updated the following in the struct sgdma_instance_t table:
	  • Updated values in Structure Member column.
	  • Added axi4_mm_data_width, num_of_s2mm_desc, *per_s2mm_desc_length, and fp rows.
	  • Removed **buffer* row.

API Variables	Updated Variable Name column to add *_ptr* to the variable name.

Revision 1.1, January 2024

Section	Change Summary
Inclusive Language	Added this section.
Introduction	Updated the IP Version to 2.0.1 in 1.4. Driver and IP Compatibility section.
API Description	• Updated header name for 2.1. sgdma_init() – 2.10. sgdma_register_write() sections. • Added the full API descriptions into respective paragraph for 2.1. sgdma_init() – 2.10. sgdma_register_write() sections. • Updated the descriptions for *this_sgdma, base_addr, and num_of_desc parameters in 2.1. sgdma_init() section. • Updated the paragraph to: This API configures the single or multiple buffer descriptors depending on the input parameters, triggers the DMA, and waits for completion depending on the blocking parameters in 2.7. mm2s_buf_desc_dma() and 2.8. s2mm_buf_desc_dma() sections. • Added 2.9. sgdma_register_read() and 2.10. sgdma_register_write() sections.
Function Call Flow Diagrams	• Updated header name for 3.1. sgdma_init() – 3.10. sgdma_register_write() sections. • Updated Figure 3.1. sgdma_init() – Figure 3.8. s2mm_buf_desc_dma(). • Added 3.9. sgdma_register_read() and 3.10. sgdma_register_write() sections.
API Data Structure and Enums	• Added and Enums to header name of 4. API Data Structure and Enums section. • Added 4.12. struct sgdma_ctrl_reg_t – 4.17. enum sgdma_reg_type_t sections.
API Macros	• Updated macro names from *FAILURE* to *RET_FAILURE* and *SUCCESS* to *RET_SUCCESS*. • Removed *DWORD_4* macro. • Added *IS_NULL* macro.

Revision 1.0, December 2023

Section	Change Summary
All	Initial release.

www.latticesemi.com