10G Ethernet Driver API Reference

Technical Note

FPGA-TN-02375-1.0

July 2024

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.

Abbreviations	Definition
AHB	Advance High-Performance Bus
APB	Advanced Peripheral Bus
AXI	Advanced eXtensible Interface
CRC	Cyclic Redundancy Check
IEEE	Institute of Electrical and Electronics Engineers
MAC	Media Access Controller
RX	Receive
TX	Transmit
XG	Ten Gigabit

1. Introduction

The 10Gb Ethernet Media Access Controller IP core is a complex core containing all the necessary logic and interfacing and clocking infrastructures to integrate an external industry-standard Ethernet PHY with an internal processor efficiently with minimal overhead.

The 10Gb MAC IP core supports the ability to transmit and receive data between standard interfaces such as APB, AHB-Lite or AXI4-Lite, and an Ethernet network. The main function of the 10Gb Ethernet IP is to ensure that the media access rules specified in the IEEE 802.3-2012 standard are met while transmitting a frame of data over the Ethernet. On the receiving side, the 10Gb MAC extracts different components of a frame and transfers them to higher applications through the AXI4-Stream interface.

For more information about the IP core, refer to the 10G Ethernet MAC + PHY IP User Guide (FPGA-IPUG-02245).

1.1. Purpose

The 10Gb MAC and its SDK is a set of application programming interfaces (APIs) that provide access to specific Lattice hardware and software capabilities. This document is a reference guide for developers that provides 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 CertusPro™-NX and Lattice Avant™ devices. This technical guide assumes that you have expertise in embedded systems and FPGA technologies.

1.3. Driver Version

XG_ETHERNET_CONTROLLER 24.01.00

1.4. Driver and IP Compatibility

Table 1.1. Driver and IP Compatibility

Driver Version	IP Version
24.01.00	2.1.x 3.1.x

Table 1.2. Quick Facts on Driver Test Environment

Hardware Device	Tool Version
LAV-AT-G70 device (MAC Only Simulation Mode)	Lattice Propel™ Builder 2024.1
	Lattice Propel SDK 2024.1
	Lattice Radiant™ software 2024.1
	Radiant Programmer 2024.1

2. API Description

This section describes the APIs for the 10Gb Ethernet IP core.

2.1. xg_ethernet_init()

This API is used to initialize the base address of the 10Gb Ethernet IP base address and pass the IP capabilities configuration for the driver.

```
int xg_ethernet_init(xg_ethernet_instance *this_xg_ethernet, xg_ethernet_config *config)
```

In/Out	Parameter	Description	Returns
In	this_xg_ethernet	A handle that stores information in the xg_ethernet_instance structure.	0: success 1: failure
In	config	Base address and IP capabilities, which are configured in the Lattice Propel™ Builder software is assigned to the controller.	

2.2. xg_ethernet_start()

This API is used to activate the TX MAC and RX MAC functionalities for the 10Gb Ethernet IP core.

```
int xg_ethernet_start(xg_ethernet_instance *this_xg_ethernet)
```

In/Out	Parameter	Description	Returns
In	this_xg_ethernet	A handle that stores information in the xg_ethernet_instance structure.	0: success 1: failure

2.3. xg_ethernet_stop()

This API is used to deactivate the TX MAC and RX MAC functionalities for the 10Gb Ethernet IP core.

```
int xg_ethernet_stop(xg_ethernet_instance *this_xg_ethernet)
```

In/Out	Parameter	Description	Returns
In	this_xg_ethernet	A handle that stores information in the xg_ethernet_instance structure.	0: success 1: failure

2.4. xg_ethernet_reset()

This API is used to deactivate the TX MAC and RX MAC of the 10Gb Ethernet IP core and reinitialize the xg_ethernet_instance parameter with new xg_ethernet_config.

```
int xg_ethernet_reset(xg_ethernet_instance *this_xg_ethernet, xg_ethernet_config
*config)
```

In/Out	Parameter	Description	Returns
In	this_xg_ethernet	A handle that stores information in the xg_ethernet_instance structure.	0: success 1: failure
In	config	Base address and IP capabilities, which are configured in the Propel Builder software are assigned to the controller.	

2.5. xg_ethernet_set_mac_options()

This API is used to enable the TX and RX functionalities of the 10Gb Ethernet IP control option and the mac_cfg_options.

```
int xg_ethernet_set_mac_options(xg_ethernet_instance *this_xg_ethernet, unsigned int options)
```

In/Out	Parameter	Description	Returns
In	this_xg_ethernet	A handle that stores information in the xg_ethernet_instance structure.	0: success 1: failure
In	options	TX control options and RX control options.	

2.6. xg_ethernet_clear_mac_options()

This API is used to disable the TX and RX functionalities of the 10Gb Ethernet IP control option and the mac_cfg_options.

```
int xg_ethernet_clear_mac_options(xg_ethernet_instance *this_xg_ethernet, unsigned int options)
```

In/Out	Parameter	Description	Returns
In	this_xg_ethernet	A handle that stores information in the xg_ethernet_instance structure.	0: success 1: failure
In	options	TX control options and RX control options.	

2.7. xg_ethernet_set_mac_address()

This API is used to set the MAC address of the 10Gb Ethernet IP core.

```
int xg_ethernet_set_mac_address(xg_ethernet_instance *this_xg_ethernet, char *mac_address)
```

In/Out	Parameter	Description	Returns
In	this_xg_ethernet	A handle that stores information in the xg_ethernet_instance structure.	0: success 1: failure
In	mac_address	MAC address in char[6] format. For example: if the MAC address is AC-DE-48-00-00-80, arrange it in char mac_address = {AC, DE, 48, 00, 00, 80};	

2.8. xg_ethernet_get_mac_address()

This API is used to get the MAC address of the 10Gb Ethernet IP core.

```
int xg_ethernet_get_mac_address(xg_ethernet_instance *this_xg_ethernet, char *mac_address)
```

In/Out	Parameter	Description	Returns
In	this_xg_ethernet	A handle that stores information in the xg_ethernet_instance structure.	0: success

In/Out	Parameter	Description	Returns
Out	mac_address	MAC address in char[6] format. For example: if the obtained mac_address = {AC, DE, 48, 00, 00, 80}, it is equivalent to AC-DE-48-00-00-80.	1: failure

2.9. xg_ethernet_conv_mac_to_crc()

This API is used to calculate cyclic redundancy check (CRC) from the MAC address.

```
int xg_ethernet_conv_mac_to_crc(char *mac_address)
```

In/Out	Parameter	Description	Returns
In	mac_address	MAC address in char[6] format. For example: if the MAC address is AC-DE-48-00-00-80, arrange it in char mac_address = {AC, DE, 48, 00, 00, 80};	CRC

2.10. xg_ethernet_set_multicast_filter()

This API is used to set multicast filter based on the MAC address.

```
int xg_ethernet_set_multicast_filter(xg_ethernet_instance *this_xg_ethernet, char
*mac_address)
```

In/Out	Parameter	Description	Returns
In	this_xg_ethernet	A handle that stores information in the xg_ethernet_instance structure.	0: success 1: failure
In	mac_address	MAC address in char[6] format. For example: if the MAC address is AC-DE-48-00-00-80, arrange it in char mac_address = {AC, DE, 48, 00, 00, 80};	

2.11. xg_ethernet_clear_multicast_filter()

This API is used to clear the multicast filter based on the MAC address.

```
int xg_ethernet_clear_multicast_filter(xg_ethernet_instance *this_xg_ethernet, char
*mac_address)
```

In/Out	Parameter	Description	Returns
In	this_xg_ethernet	A handle that stores information in the xg_ethernet_instance structure.	0: success 1: failure
In	mac_address	MAC address in char[6] format. For example: if the MAC address is AC-DE-48-00-00-80, arrange it in char mac_address = {AC, DE, 48, 00, 00, 80};	

2.12. xg_ethernet_get_rx_vlan_tag()

This API is used to get the RX VLAN tag field of the most recent tagged frame that was received.

```
int xg_ethernet_get_rx_vlan_tag(xg_ethernet_instance *this_xg_ethernet, unsigned int *vlan_tag)
```

In/Out	Parameter	Description	Returns
In	this_xg_ethernet	A handle that stores information in the xg_ethernet_instance structure.	0: success 1: failure
Out	vlan_tag	VLAN tag ID of the most recent tagged frame.	

2.13. xg_ethernet_get_rx_idle()

This API is used to get the status of the RX MAC.

```
int xg_ethernet_get_rx_idle(xg_ethernet_instance *this_xg_ethernet, unsigned char *rx_idle)
```

In/Out	Parameter	Description	Returns
In	this_xg_ethernet	A handle that stores information in the xg_ethernet_instance structure.	0: success 1: failure
Out	rx_idle	Status of the RX MAC. Active: 0 Inactive: 1	

2.14. xg_ethernet_get_tx_idle()

This API is used to get the status of the TX MAC.

```
int xg_ethernet_get_tx_idle(xg_ethernet_instance *this_xg_ethernet, unsigned char *tx_idle)
```

In/Out	Parameter	Description	Returns
In	this_xg_ethernet	A handle that stores information in the xg_ethernet_instance structure.	0: success 1: failure
Out	tx_idle	Status of the TX MAC. Active: 0 Inactive: 1	

2.15. xg_ethernet_set_flow_control()

This API is used to enable the flow control functionality of the TX MAC.

```
int xg_ethernet_set_flow_control(xg_ethernet_instance *this_xg_ethernet)
```

In/Out	Parameter	Description	Returns
In	this_xg_ethernet	A handle that stores information in the xg_ethernet_instance structure.	0: success 1: failure

2.16. xg_ethernet_set_rx_pause_en()

This API is used to enable the RX MAC to receive PAUSE frame.

```
int xg_ethernet_set_rx_pause_en(xg_ethernet_instance *this_xg_ethernet)
```

In/Out	Parameter	Description	Returns
In	this_xg_ethernet	A handle that stores information in the xg_ethernet_instance structure.	0: success 1: failure

2.17. xg_ethernet_set_pause_tm()

This API is used to configure the pause time for a flow control packet (sourced by the TX MAC).

```
int xg_ethernet_set_pause_tm(xg_ethernet_instance *this_xg_ethernet, int pause_time)
```

In/Out	Parameter	Description	Returns
In	this_xg_ethernet	A handle that stores information in the xg_ethernet_instance structure.	0: success 1: failure
In	pause_time	Represents a 16-bits value used for flow control packet.	

2.18. xg_ethernet_tx_pause_frm()

This API is used to enable TX MAC to transmit a PAUSE frame.

```
int xg_ethernet_tx_pause_frm(xg_ethernet_instance *this_xg_ethernet)
```

In/Out	Parameter	Description	Returns
In	this_xg_ethernet	A handle that stores information in the xg_ethernet_instance structure.	0: success 1: failure

2.19. xg_ethernet_set_ipg_val()

This API is used to configure the IPG value to be used by the TX MAC.

```
int xg_ethernet_set_ipg_val(xg_ethernet_instance *this_xg_ethernet, int ipg_val)
```

In/Out	Parameter	Description	Returns
In	this_xg_ethernet	A handle that stores information in the xg_ethernet_instance structure.	0: success 1: failure
In	ipg_value	Represents a 5-bit value used for inter-frame gap.	

2.20. xg_ethernet_ipg_strectch_mode()

This API is used to configure the TX MAC to operate in the IFG stretch mode to match the data rates of OC-192.

```
int xg_ethernet_set_ipg_stretch_mode(xg_ethernet_instance *this_xg_ethernet)
```

In/Out	Parameter	Description	Returns
In	this_xg_ethernet	A handle that stores information in the xg_ethernet_instance structure.	0: success 1: failure

2.21. xg_ethernet_get_statistic_counter()

This API is used to read a value from specific statistic counter register.

```
int xg_ethernet_get_statistic_counters(xg_ethernet_instance *this_xg_ethernet, unsigned int reg_offset, unsigned long long *counter_val)
```

In/Out	Parameter	Description	Returns
In	this_xg_ethernet	A handle that stores information in the xg_ethernet_instance structure.	0: success 1: failure
In	reg_offset	Statistic counter register offset.	
Out	counter_val	Value of statistic counter.	

2.22. xg_ethernet_print_statistic_counter()

This API is used to print the name of the statistic counter register string, along with its corresponding register value.

```
int xg_ethernet_print_statistic_counter(xg_ethernet_instance *this_xg_ethernet, unsigned int reg_offset)
```

In/Out	Parameter	Description	Returns
In	this_xg_ethernet	A handle that stores information in the xg_ethernet_instance structure.	0: success 1: failure
In	reg_offset	Statistic counter register offset.	

3. Function Call Flow Diagrams

This section shows the function call flow diagrams for 10Gb Ethernet IP core.

3.1. xg_ethernet_init()

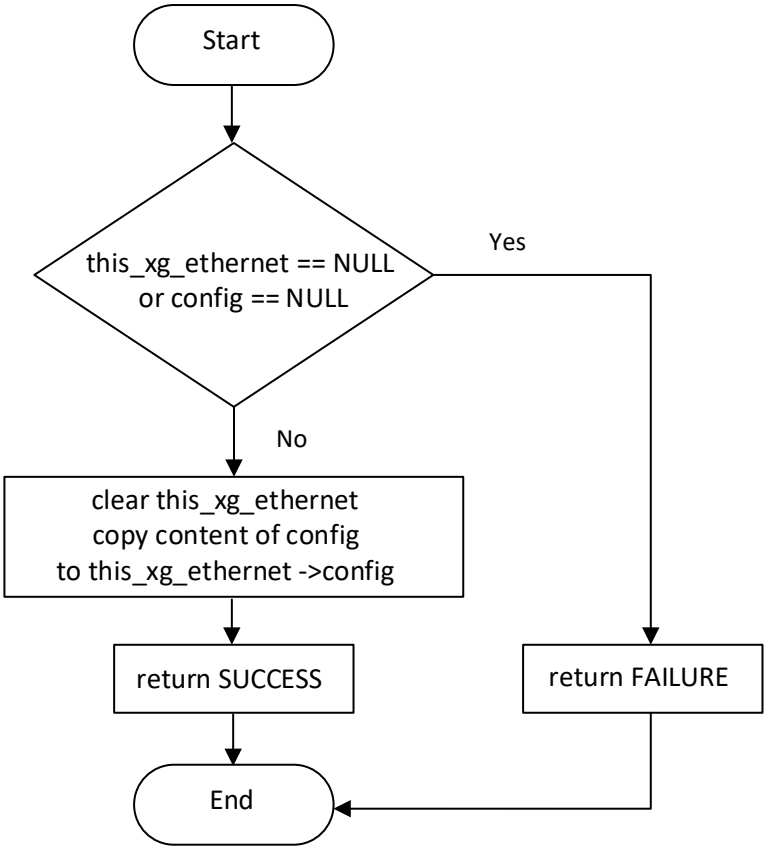

Figure 3.1. int xg_ethernet_init()

3.2. xg_ethernet_start()

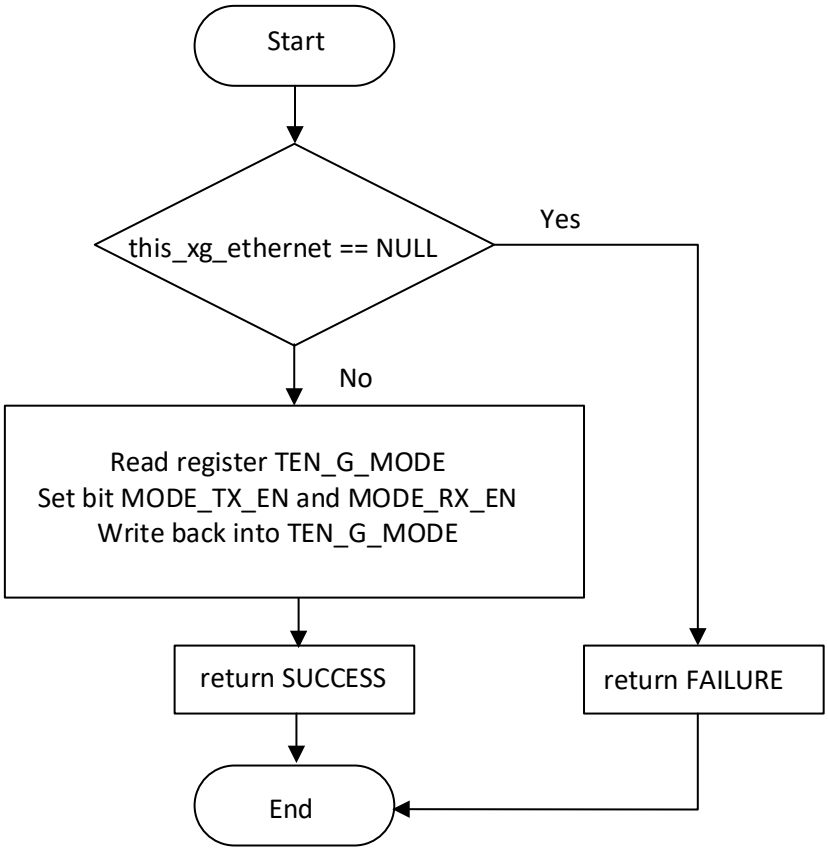

Figure 3.2. int xg_ethernet_start()

3.3. xg_ethernet_stop()

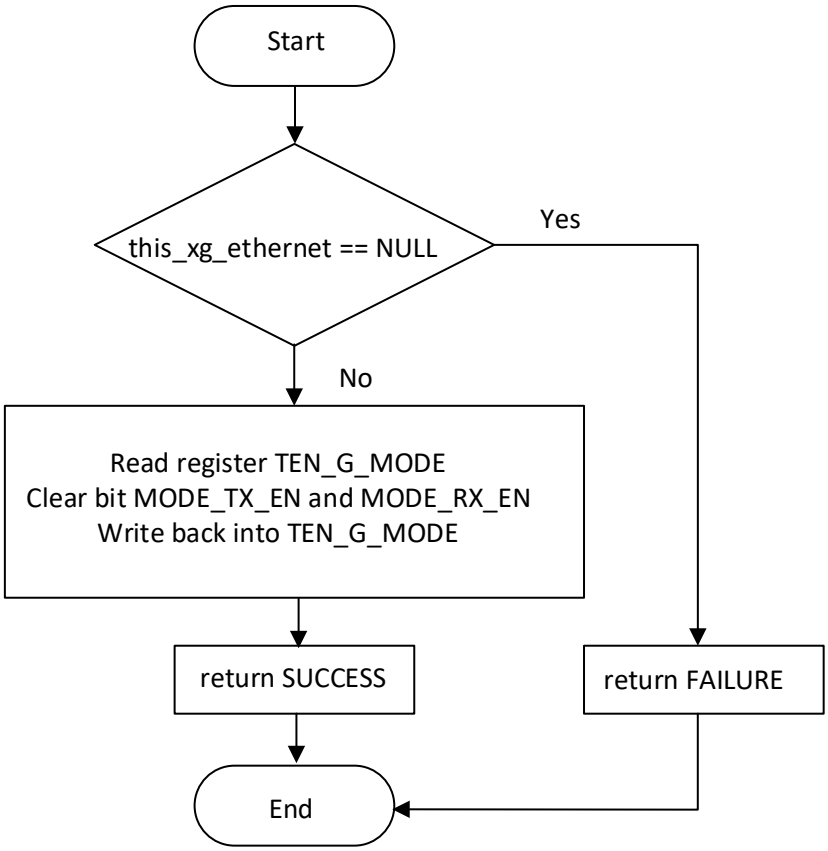

Figure 3.3. int xg_ethernet_stop()

3.4. xg_ethernet_reset()

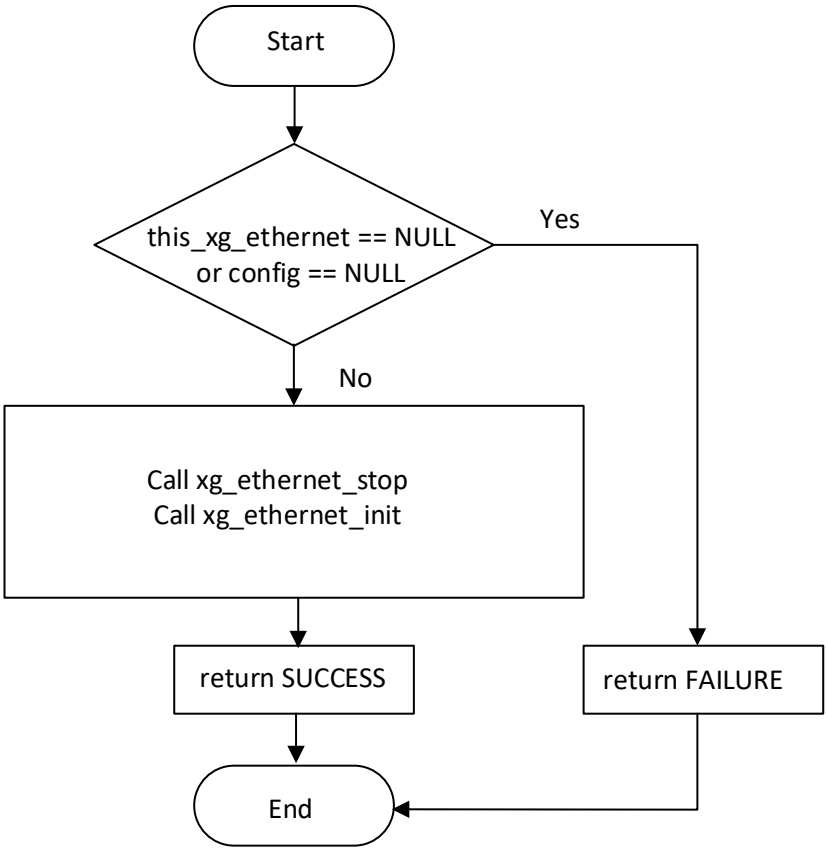

Figure 3.4. int xg_ethernet_reset()

3.5. xg_ethernet_set_mac_options()

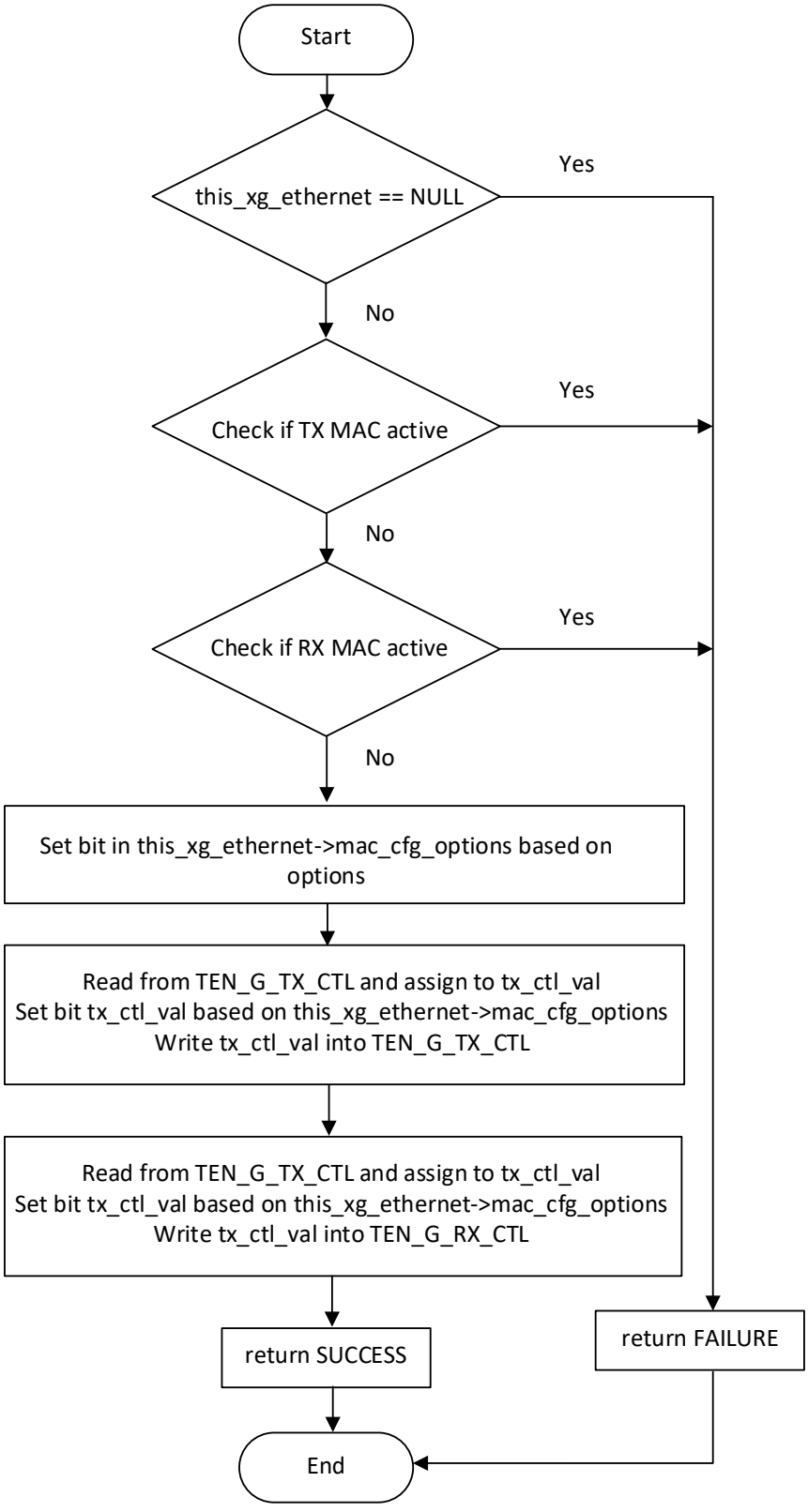

Figure 3.5. int xg_ethernet_set_mac_options()

3.6. xg_ethernet_clear_mac_options()

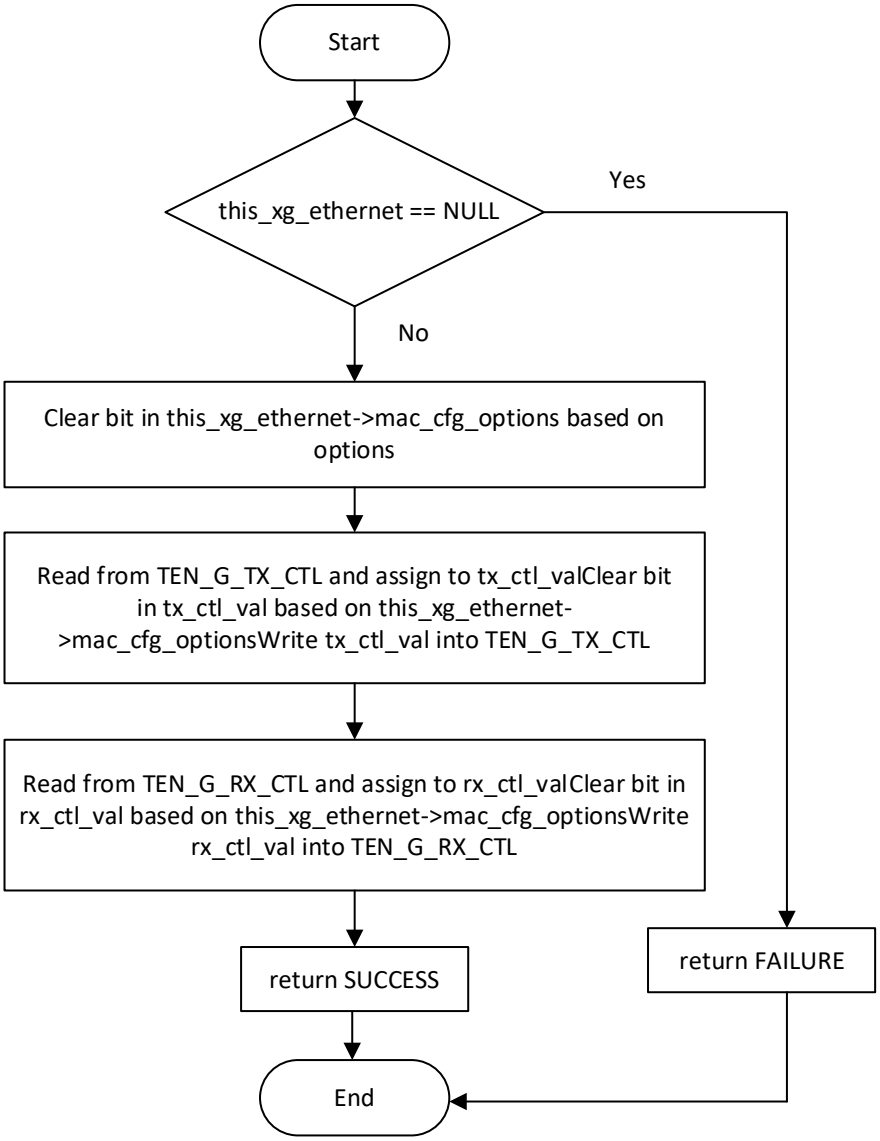

Figure 3.6. int xg_ethernet_clear_mac_options()

3.7. xg_ethernet_set_mac_address()

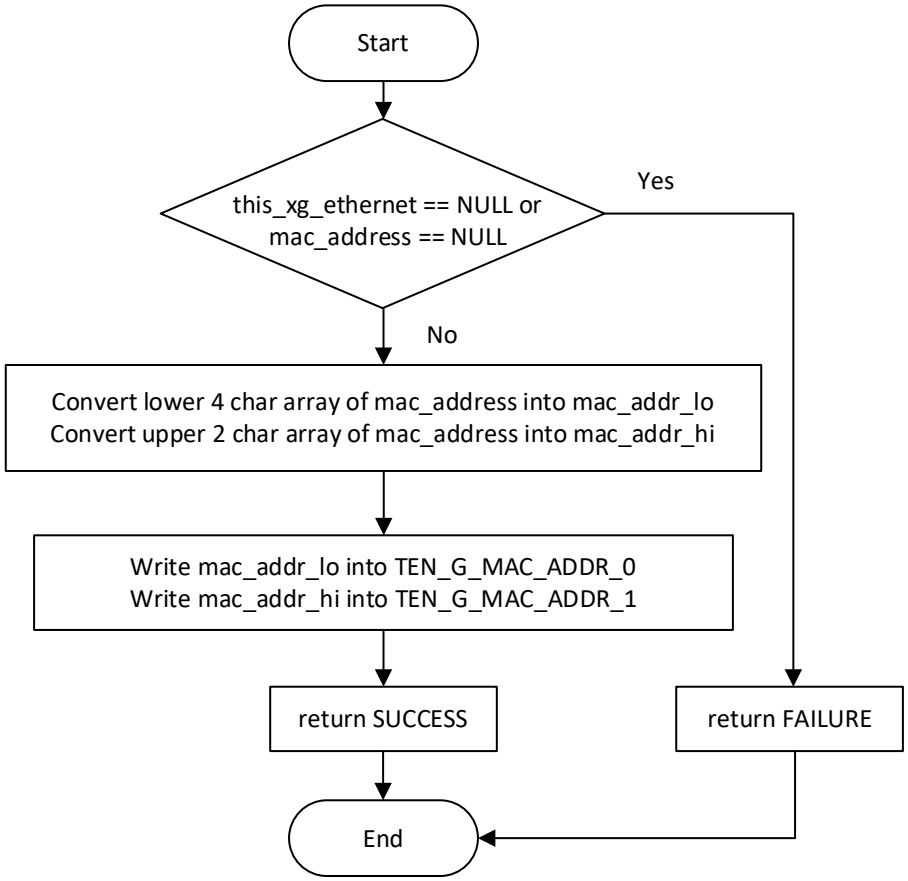

Figure 3.7. int xg_ethernet_set_mac_address()

3.8. xg_ethernet_get_mac_address()

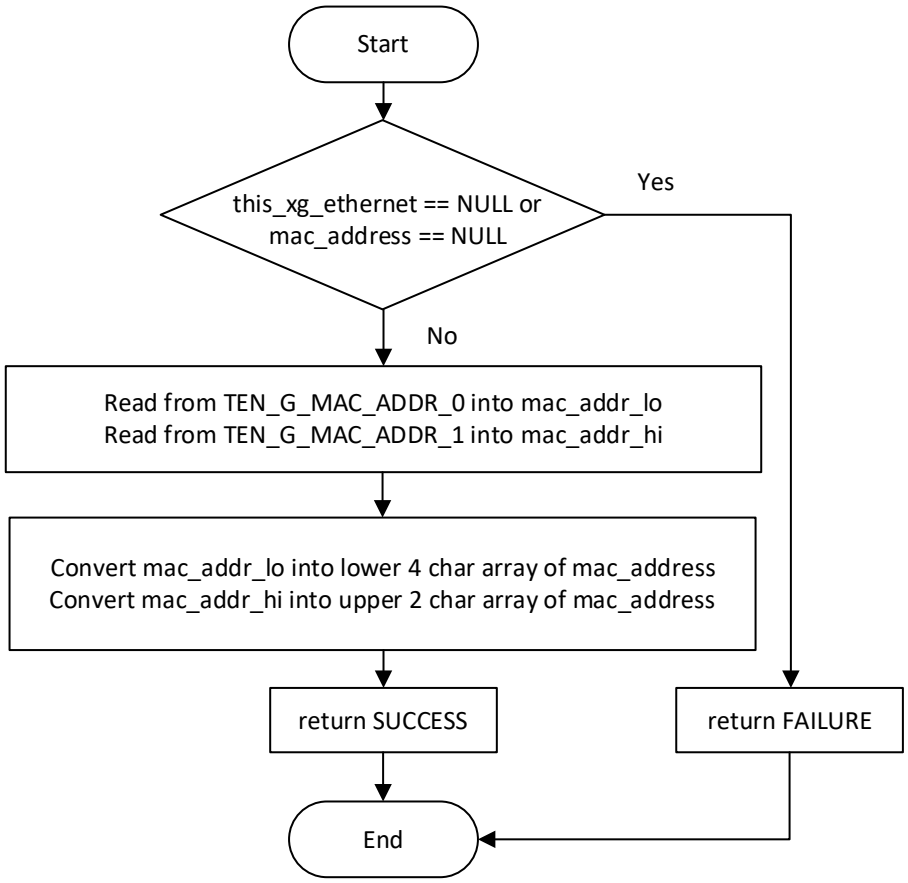

Figure 3.8. int xg_ethernet_get_mac_address()

3.9. xg_ethernet_conv_mac_to_crc()

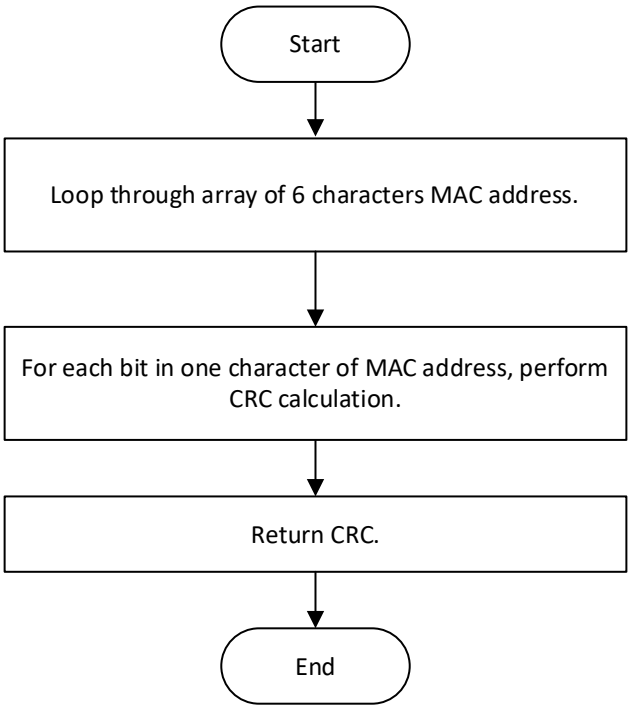

Figure 3.9. int xg_ethernet_conv_mac_to_crc()

3.10. xg_ethernet_set_multicast_filter()

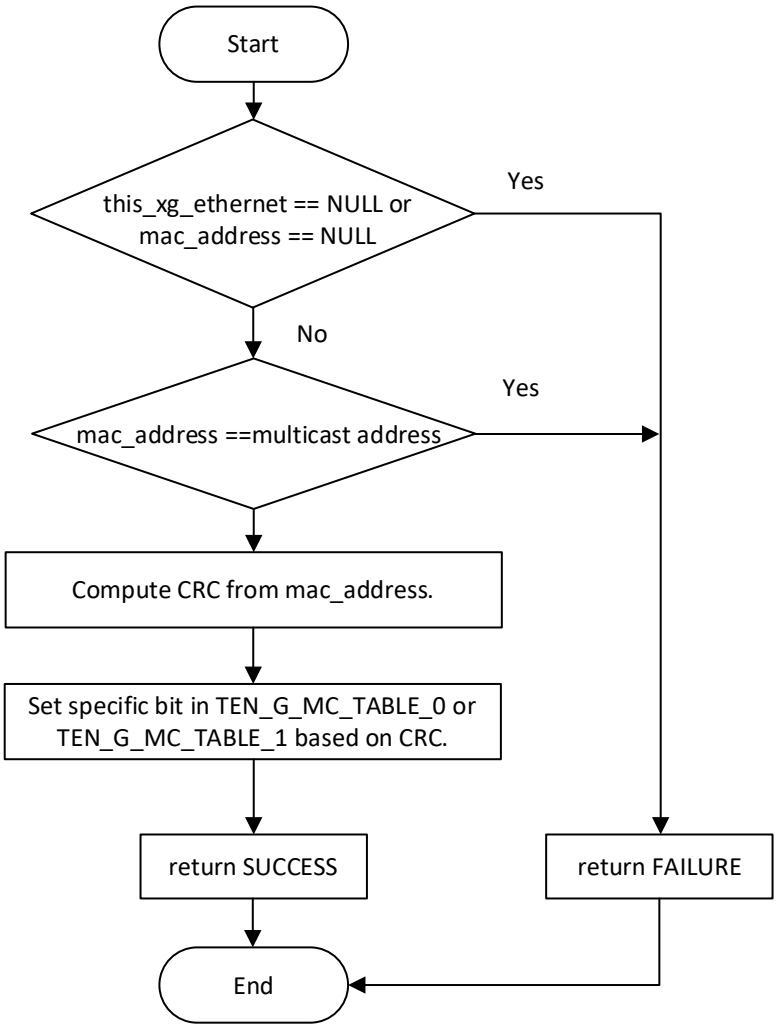

Figure 3.10. int xg_ethernet_set_multicast_filter()

3.11. xg_ethernet_clear_multicast_filter()

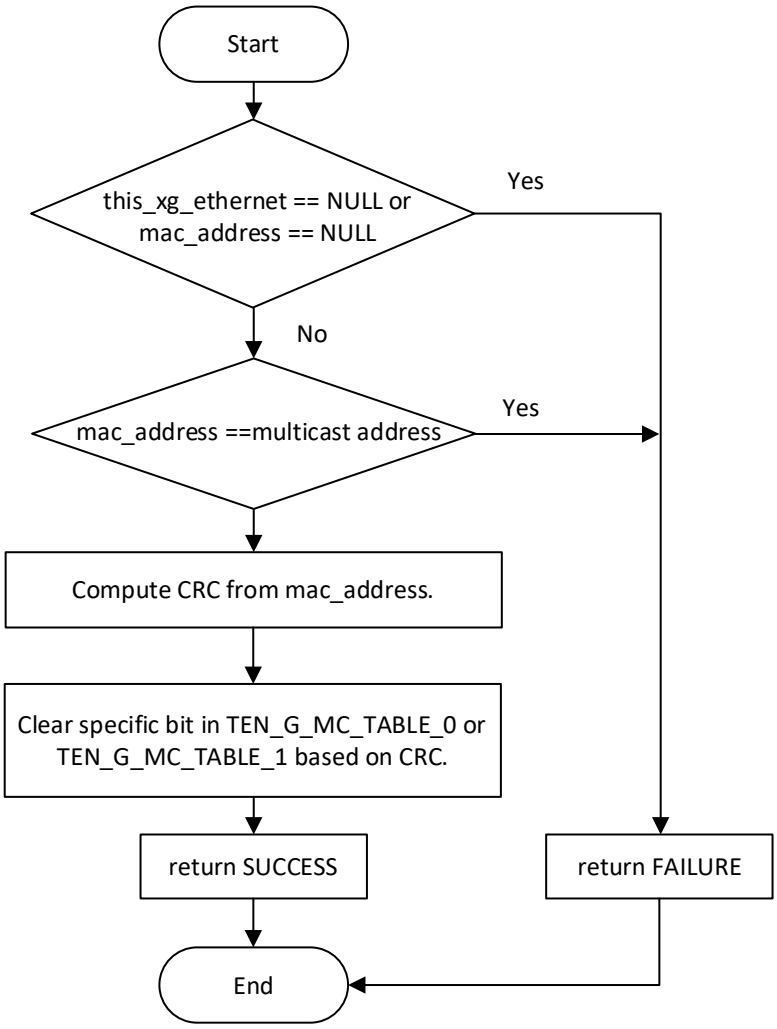

Figure 3.11. int xg_ethernet_clear_multicast_filter()

3.12. xg_ethernet_get_rx_vlan_tag()

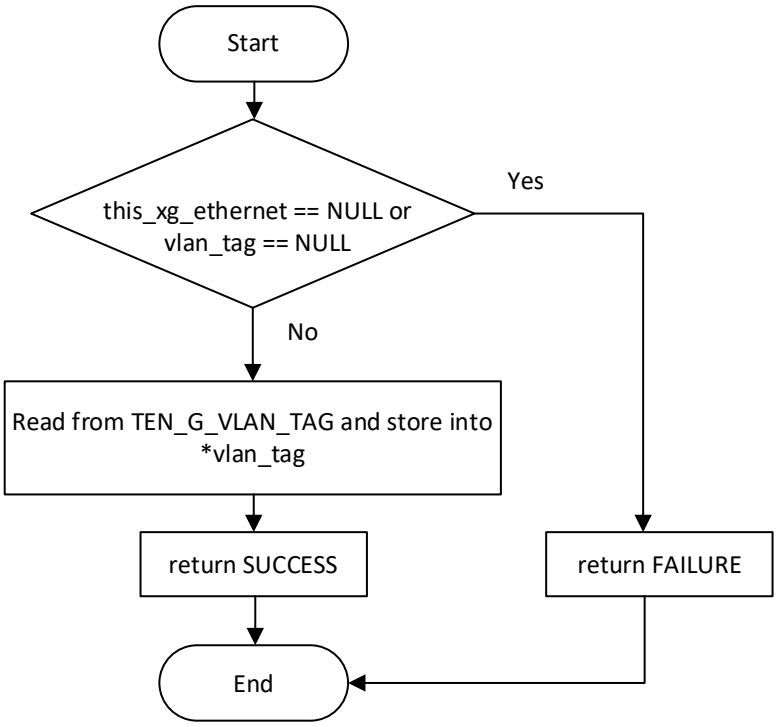

Figure 3.12. int xg_ethernet_get_rx_vlan_tag()

3.13. xg_ethernet_get_rx_idle()

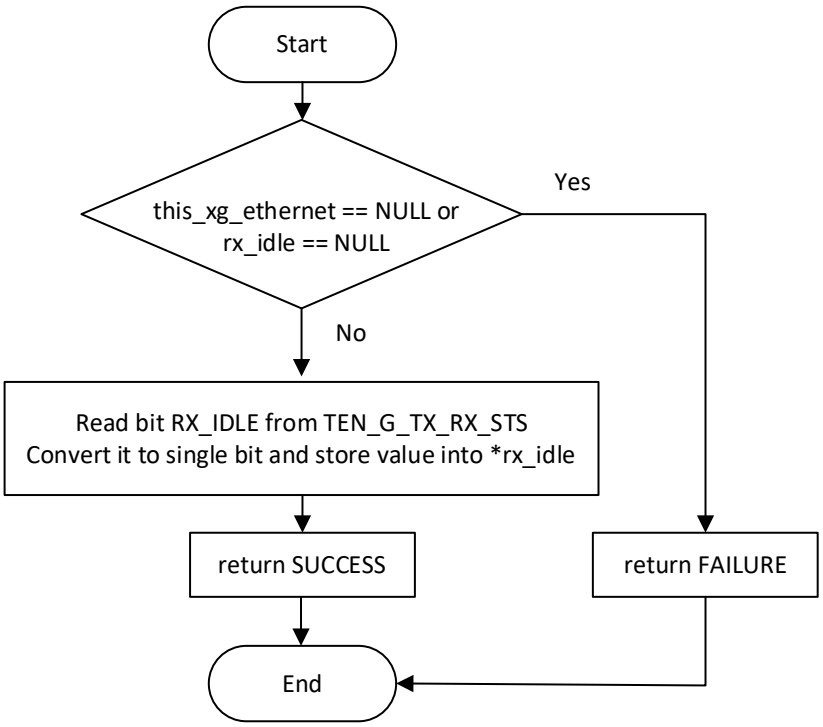

Figure 3.13. int xg_ethernet_get_rx_idle()

3.14. xg_ethernet_get_tx_idle()

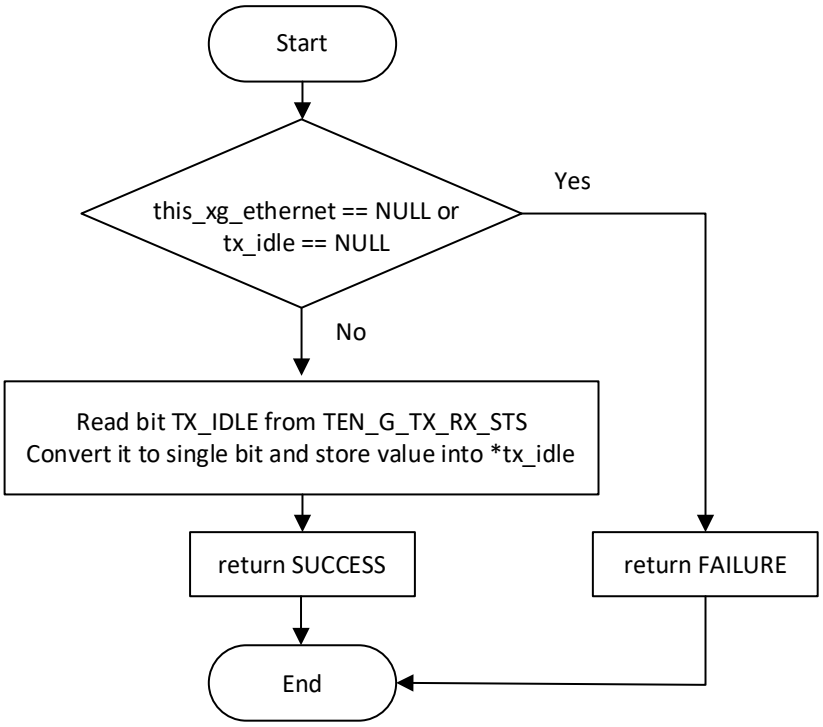

Figure 3.14. int xg_ethernet_get_tx_idle()

3.15. xg_ethernet_set_flow_control()

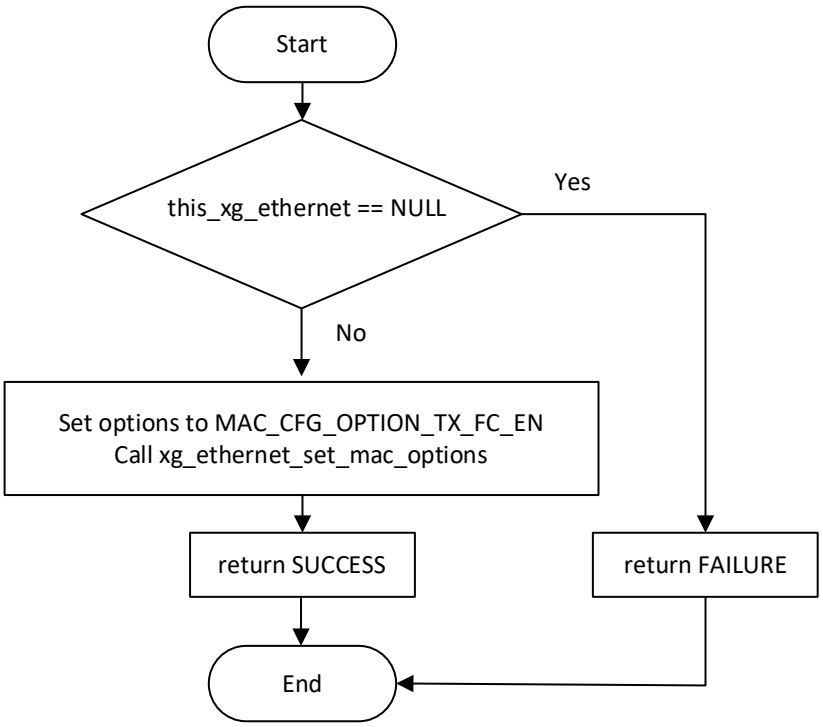

Figure 3.15. int xg_ethernet_set_flow_control()

3.16. xg_ethernet_set_rx_pause_en()

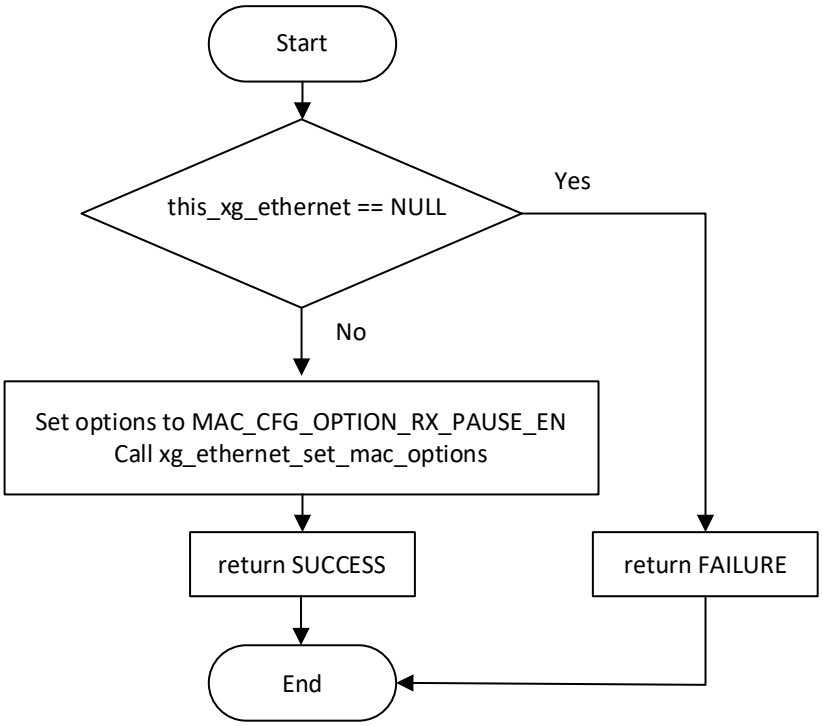

Figure 3.16. int xg_ethernet_set_rx_pause_en()

3.17. xg_ethernet_set_pause_tm()

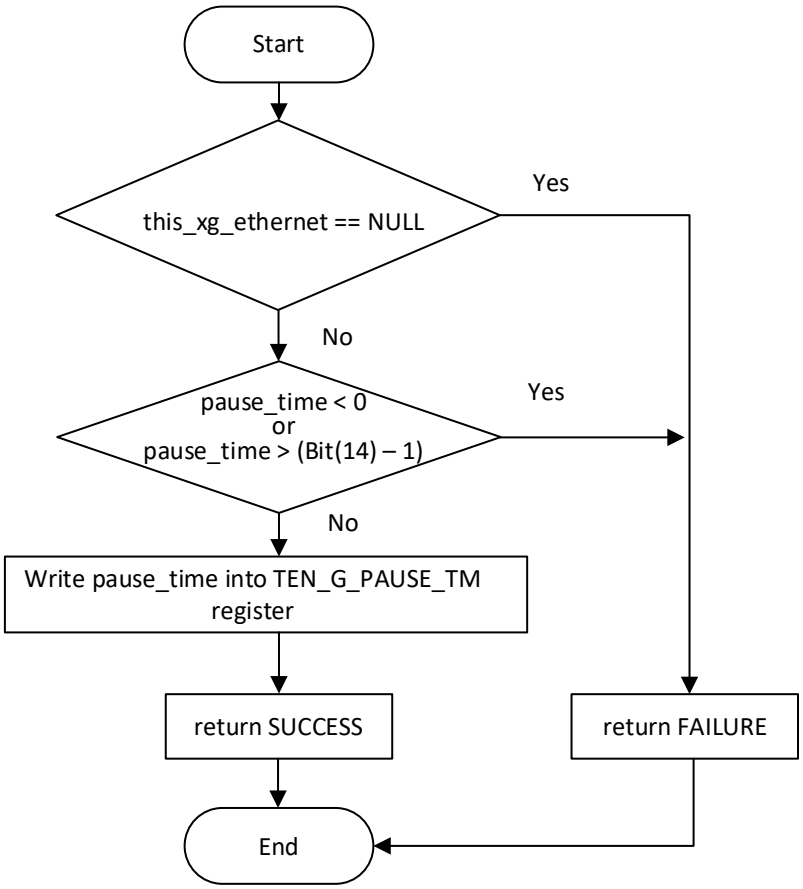

Figure 3.17. int xg_ethernet_set_pause_tm()

3.18. xg_ethernet_tx_pause_frm()

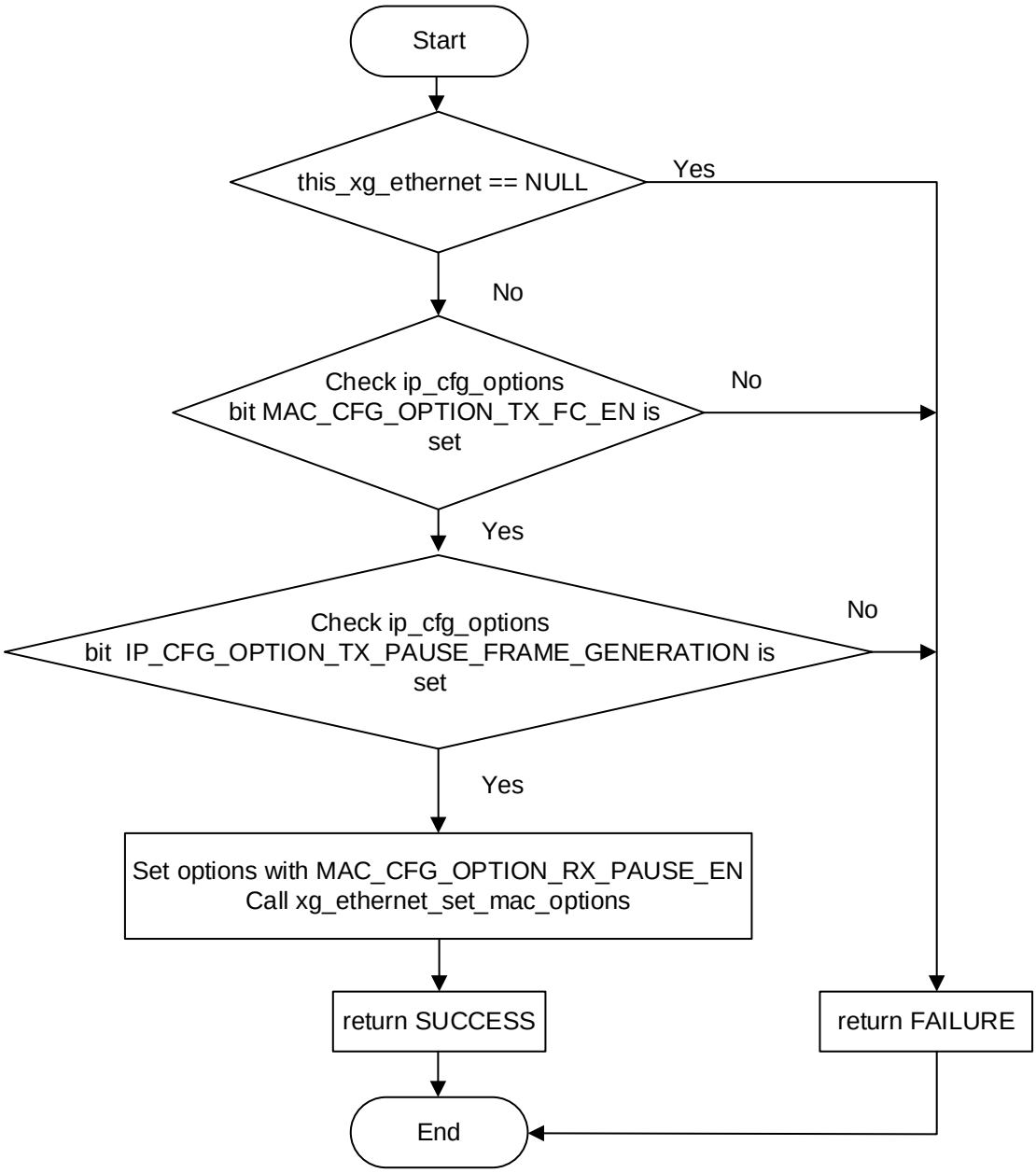

Figure 3.18. int xg_ethernet_tx_pause_frm()

3.19. xg_ethernet_set_ipg_val()

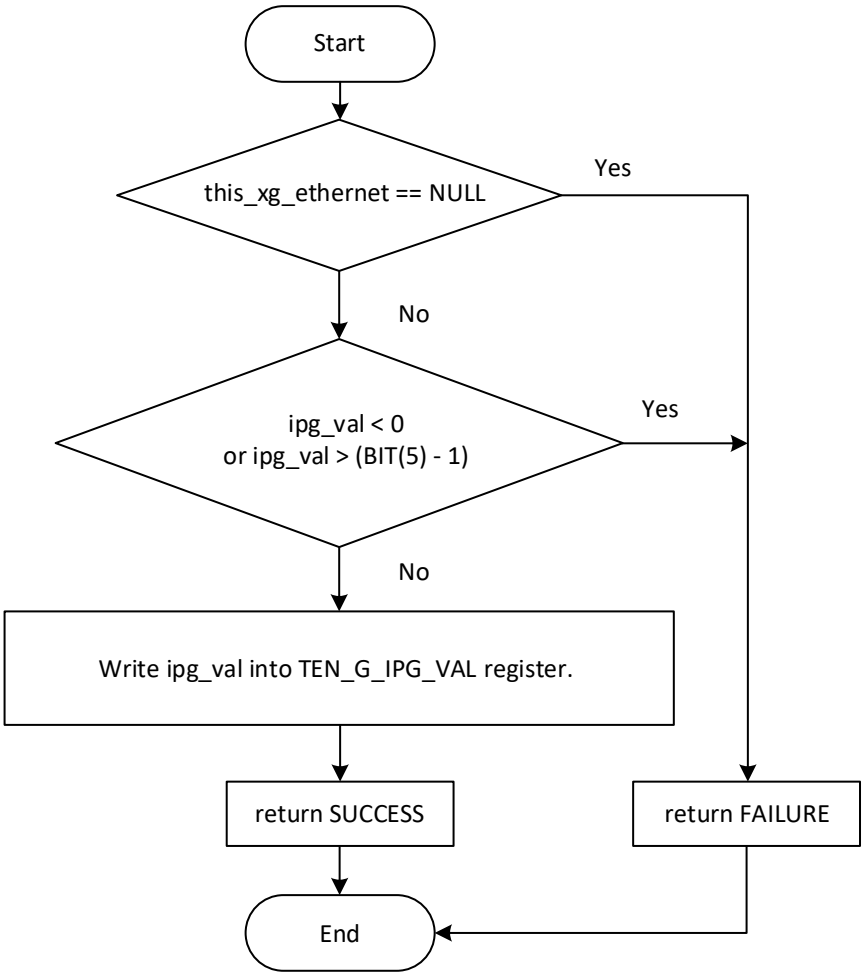

Figure 3.19. int xg_ethernet_set_ipg_val()

3.20. xg_ethernet_set_ipg_stretch_mode()

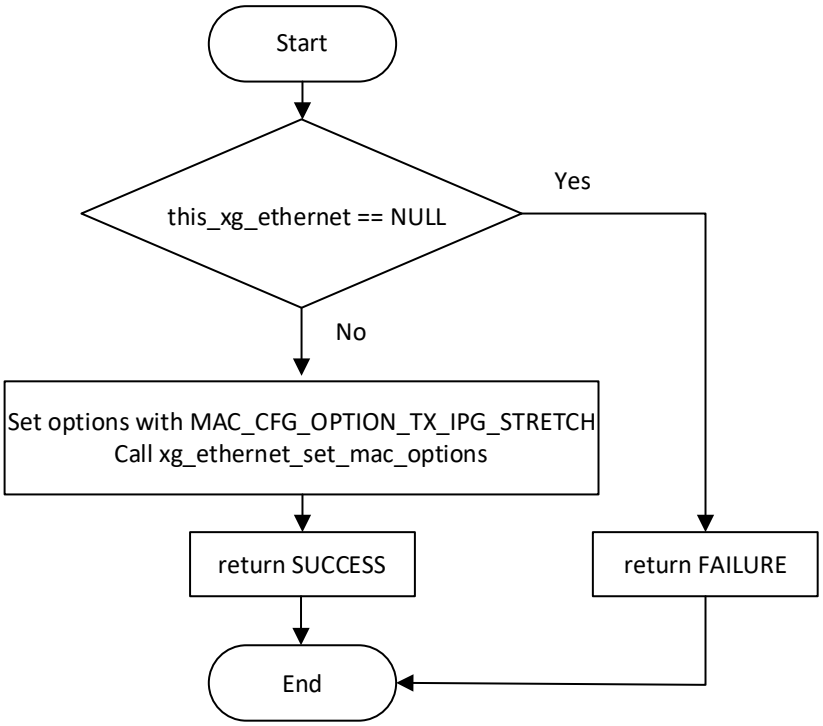

Figure 3.20. int xg_ethernet_set_ipg_stretch_mode()

3.21. xg_ethernet_get_statistic_counter()

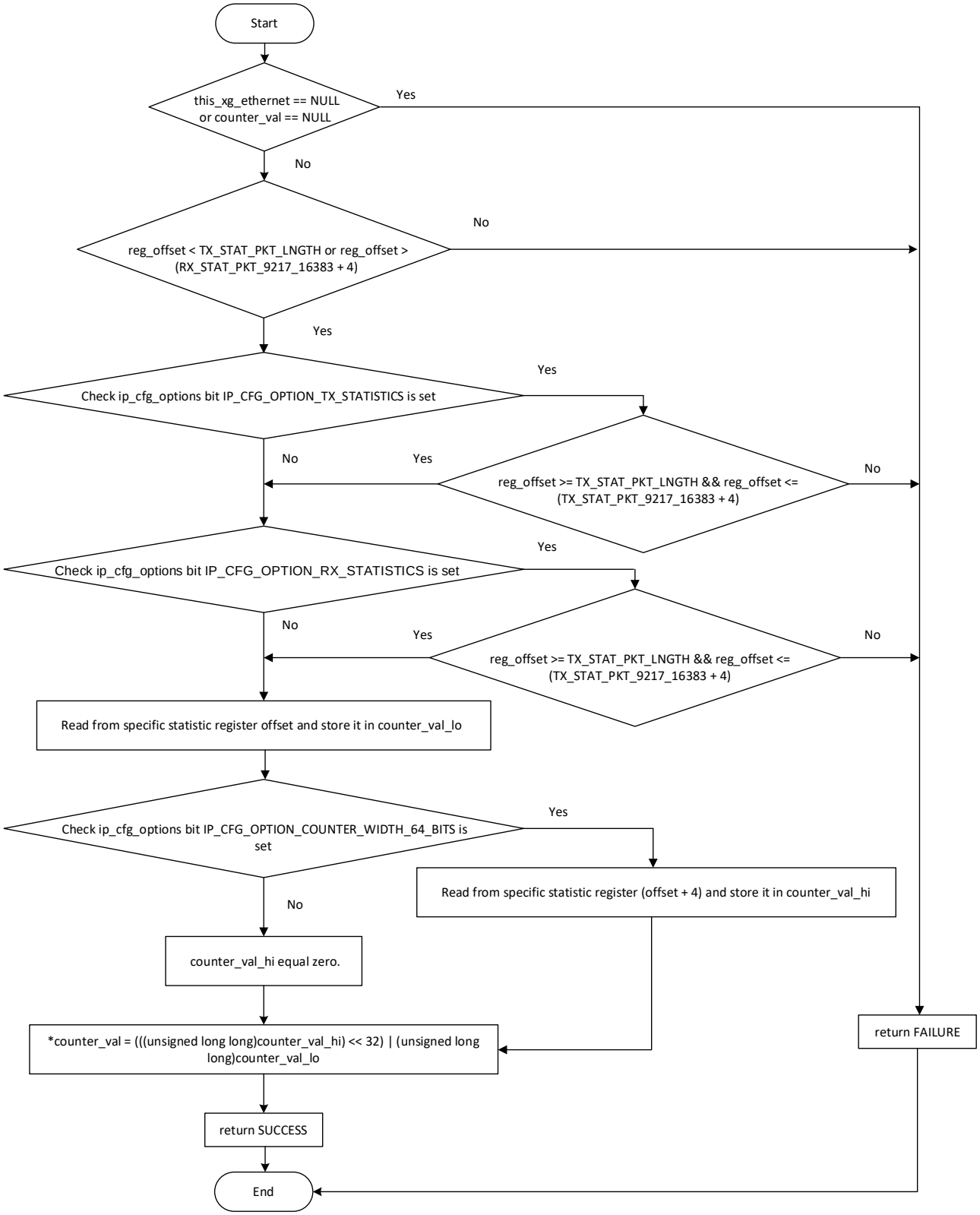

Figure 3.21. int xg_ethernet_get_statistic_counter()

3.22. xg_ethernet_print_statistic_counter()

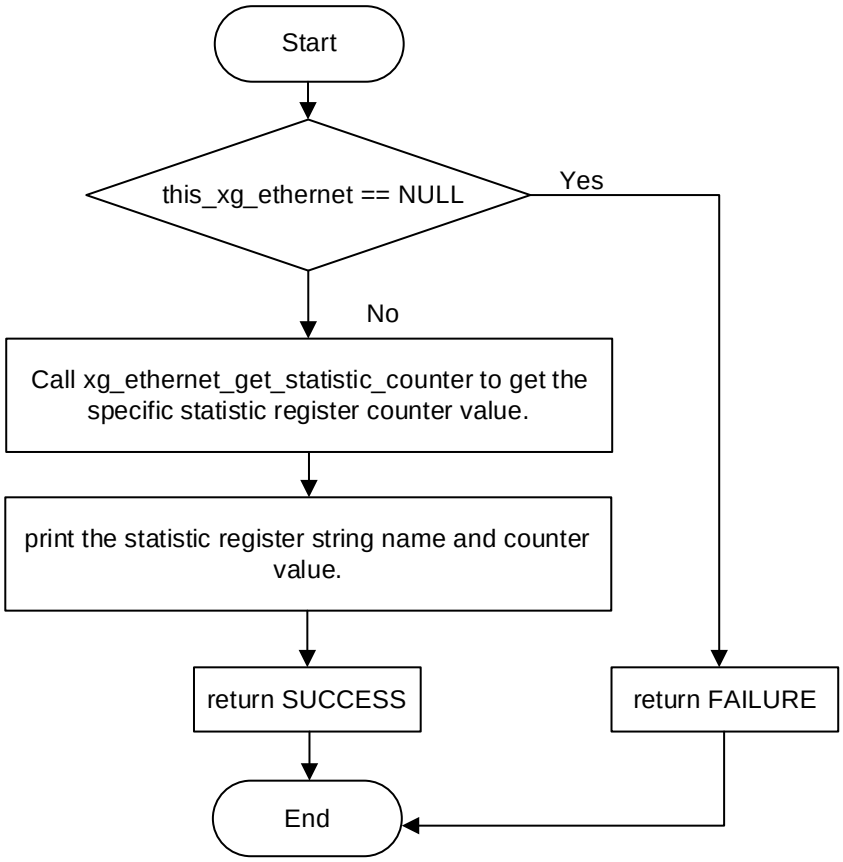

Figure 3.22. int xg_ethernet_print_statistic_counter()

4. API Data Structures

4.1. struct xg_ethernet_config

Table 4.1. xg_ethernet_config Parameters

Data Type	Struct Member	Description
unsigned int	base_address	Base address of the 10Gb Ethernet IP.
unsigned int	ip_cfg_options	IP capabilities configured in the Propel Builder.

4.2. struct xg_ethernet_instance

Table 4.2. xg_ethernet_instance Parameters

Data Type	Struct Member	Description
xg_ethernet_config	config	Configuration information of the 10Gb Ethernet IP.
unsigned char	phy	Physical layer information.
unsigned int	mac_cfg_options	Configuration settings for TX and RX MAC.

5. API Macros

Table 5.1. XG_ETHERNET API Macros Description

Macro	Description
#define XG_ETHERNET_CONTROLLER_DRV_VER	XG Ethernet driver version.
#define TEN_G_MODE	Transmit MAC control register.
#define TEN_G_TX_CTL	Receive MAC control register.
#define TEN_G_RX_CTL	Maximum packet size register.
#define TEN_G_MAX_PKT_LNGTH	IPG value register.
#define TEN_G_IPG_VAL	MAC address register word 0.
#define TEN_G_MAC_ADDR_0	MAC address register word 1.
#define TEN_G_MAC_ADDR_1	Transmit and receive status register.
#define TEN_G_TX_RX_STS	VLAN tag register.
#define TEN_G_VLAN_TAG	Multicast tables register word 0.
#define TEN_G_MC_TABLE_0	Multicast tables register word 1.
#define TEN_G_MC_TABLE_1	Pause opcode.
#define TEN_G_PAUSE_OPCODE	MAC control register.
#define TEN_G_MAC_CTL	PAUSE time register
#define TEN_G_PAUSE_TM	Transmit MAC control register.
#define MODE_TX_EN	TX MAC enable.
#define MODE_RX_EN	RX MAC enable.
#define TX_PASS_FCS	In-band FCS enable.
#define TX_FC_EN	Flow-control enable.
#define TX_IPG_STRETCH	IFG stretch mode.
#define TX_TRANSMIT_SHORT	Transmit short frames
#define TX_PASS_PREARM	Transmit custom preamble mode.
#define RX_PRMS	Promiscuous mode.
#define RX_PASS_FCS	In-band FCS passing.
#define RX_PAUSE_EN	Receive PAUSE frames.
#define RX_ALL_MC	Receive Multicast frames.
#define RX_BC	Receive Broadcast frames.
#define RX_SHORT	Receive Short frames.
#define RX_DROP_MAC_CTRL	Drop MAC Control frames.
#define RX_PASS_PREARM	Receive custom preamble mode.
#define RX_IDLE	Receive MAC idle.
#define TX_IDLE	Transmit MAC idle.
#define TX_PAUSE_FRAME	Transmit PAUSE frame.
#define MAC_CFG_OPTION_TX_PASS_PREARM	User option to enable transmit custom preamble mode.
#define MAC_CFG_OPTION_TX_SHORT	User option to enable transmit short frames.
#define MAC_CFG_OPTION_TX_IPG_STRETCH	User option to enable IFG stretch mode.
#define MAC_CFG_OPTION_TX_FC_EN	User option to enable flow-control.
#define MAC_CFG_OPTION_TX_PASS_FCS	User option to enable in-band FCS.
#define MAC_CFG_OPTION_RX_PASS_REARM	User option to enable receive custom preamble mode.
#define MAC_CFG_OPTION_DROP_MAC_CTRL	User option to enable Drop MAC control frames.
#define MAC_CFG_OPTION_RX_SHORT	User option to enable receive Short frames.
#define MAC_CFG_OPTION_RX_BC	User option to enable receive Broadcast frames.
#define MAC_CFG_OPTION_RX_ALL_MC	User option to enable receive Multicast frames.
#define MAC_CFG_OPTION_RX_PAUSE_EN	User option to enable receive PAUSE frames.
MAC_CFG_OPTION_RX_PASS_FCS	User option to enable receive in-band FCS passing.

Macro	Description
MAC_CFG_OPTION_PRMS	User option to enable promiscuous mode.
IP_CFG_OPTION_PHY_XGMII	IP configurations choose phy XGMII.
IP_CFG_OPTION_PHY_8_BITS_GMII	IP configurations choose phy 8 bits GMII.
IP_CFG_OPTION_PHY_16_BITS_GMII	IP configurations choose phy 16 bits GMII.
IP_CFG_OPTION_PHY_MII	IP configurations choose phy MII.
IP_CFG_OPTION_MULTICAST_ADDRESS_FILTERING	IP configurations enable multicast address filtering.
IP_CFG_OPTION_STATISTICS_COUNTER_REGISTERS	IP configurations enable statistics counter registers.
IP_CFG_OPTION_TX_PAUSE_FRAME_GENERATION	IP configurations enable PAUSE frame generation.
IP_CFG_OPTION_COUNTER_WIDTH_32_BITS	IP configurations choose statistic counter 32 bits width.
IP_CFG_OPTION_COUNTER_WIDTH_64_BITS	IP configurations choose statistic counter 64 bits width.
IP_CFG_OPTION_TX_STATISTICS	IP configurations enable transmit statistics counter.
IP_CFG_OPTION_RX_STATISTICS	IP configurations enable receive statistics counter.
SUCCESS	1
FAILURE	0

Table 5.2. XG_ETHERNET_HW API Macros Description

Macro	Description
#define TX_STAT_PKT_LNGTH	Transmit Packet Length Statistics Counter.
#define TX_STAT_ERR	Transmit TX Error Statistics Counter.
#define TX_STAT_UNDER_RUN	Transmit Underrun Error Statistics Counter.
#define TX_STAT_CRC_ERR	Transmit CRC Error Statistics Counter.
#define TX_STAT_LNGTH_ERR	Transmit Length Error Statistics Counter.
#define TX_STAT_LNG_PKT	Transmit Long packet Statistics Counter.
#define TX_STAT_MULTCST	Transmit Multicast Packet Statistics Counter.
#define TX_STAT_BRDCST	Transmit Broadcast Packet Statistics Counter.
#define TX_STAT_CNT	Transmit Control Packet Statistics Counter.
#define TX_STAT_JMBO	Transmit Jumbo Packet Statistics Counter.
#define TX_STAT_PAUSE	Transmit Pause Packet Statistics Counter.
#define TX_STAT_VLN_TG	Transmit VLAN Tag Statistics Counter.
#define TX_STAT_PKT_OK	Transmit Packet OK Statistics Counter.
#define TX_STAT_PKT_64	Transmit Packet 64 Statistics Counter.
#define TX_STAT_PKT_65_127	Transmit Packet 65 - 127 Statistics Counter.
#define TX_STAT_PKT_128_255	Transmit Packet 128-255 Statistics Counter.
#define TX_STAT_PKT_256_511	Transmit Packet 256-511 Statistics Counter.
#define TX_STAT_PKT_512_1023	Transmit Packet 512-1023 Statistics Counter.
#define TX_STAT_PKT_1024_1518	Transmit Packet 1024-1518 Statistics Counter.
#define TX_STAT_PKT_1518	Transmit Packet 1518 Statistics Counter.
#define TX_STAT_FRM_ERR	Transmit Frame Error Statistics Counter.
#define TX_STAT_PKT_1519_2047	Transmit Packet 1519-2047 Statistics Counter.
#define TX_STAT_PKT_2048_4095	Transmit Packet 2048-4095 Statistics Counter.
#define TX_STAT_PKT_4096_9216	Transmit Packet 4096-9216 Statistics Counter.
#define TX_STAT_PKT_9217_16383	Transmit Packet 9217-16383 Statistics Counter.
#define RX_STAT_PKT_LNGTH	Receive Packet Length Statistics Counter.
#define RX_STAT_VLN_TG	Receive VLAN Tag Statistics Counter.
#define RX_STAT_PAUSE	Receive Pause Packet Statistics Counter.
#define RX_STAT_FLT	Receive Filtered Packet Statistics Counter.
#define RX_STAT_UNSP_OPCODE	Receive Unsupported Opcode Statistics Counter.

Macro	Description
#define RX_STAT_BRDCST	Receive Broadcast Packet Statistics Counter.
#define RX_STAT_MULTCST	Receive Multicast Packet Statistics Counter.
#define RX_STAT_LNGTH_ERR	Receive Length Error Statistics Counter.
#define RX_STAT_LNG_PKT	Receive Long Packet Statistics Counter.
#define RX_STAT_CRC_ERR	Receive CRC Error Statistics Counter.
#define RX_STAT_PKT_DISCARD	Receive Packet Discard Statistics Counter.
#define RX_STAT_PKT_IGNORE	Receive Packet Ignored Statistics Counter.
#define RX_STAT_PKT_FRAGMENTS	Receive Packet Fragments Statistics Counter.
#define RX_STAT_PKT_JABBERS	Receive Packet Jabbers Statistics Counter.
#define RX_STAT_PKT_64	Receive Packet 64 Statistics Counter.
#define RX_STAT_PKT_65_127	Receive Packet 65-127 Statistics Counter.
#define RX_STAT_PKT_128_255	Receive Packet 128-255 Statistics Counter.
#define RX_STAT_PKT_256_511	Receive Packet 256-511 Statistics Counter.
#define RX_STAT_PKT_512_1023	Receive Packet 512-1023 Statistics Counter.
#define RX_STAT_PKT_1024_1518	Receive Packet 1024-1518 Statistics Counter.
#define RX_STAT_PKT_UNDERSIZE	Receive Packet Undersize Statistics Counter.
#define RX_STAT_PKT_UNICAST	Receive Packet Unicast Statistics Counter.
#define RX_STAT_PKT_RCVD	Packets Received Statistics Counter.
#define RX_STAT_PKT_64_GOOD_CRC	Receive Packet 64 With Good CRC Statistics Counter.
#define RX_STAT_PKT_1518_GOOD_CRC	Receive Packet 1518 With Good CRC Statistics Counter.
#define RX_STAT_PKT_1519_2047	Receive Packet 1519-2047 Statistics Counter.
#define RX_STAT_PKT_2048_4095	Receive Packet 2048-4095 Statistics Counter.
#define RX_STAT_PKT_4096_9216	Receive Packet 4096-9216 Statistics Counter.
#define RX_STAT_PKT_9217_16383	Receive Packet 9217-16383 Statistics Counter.
#define xg_ethernet_reg_name_str	A lookup table returns the string representation of the statistics counter register based on the offset of the statistics counter register.

References

- 10G Ethernet MAC + PHY IP User Guide (FPGA-IPUG-02245)
- 10Gb Ethernet MAC + PHY IP Core web page
- Lattice Radiant FPGA design software web page
- Lattice Solutions IP Cores 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, please refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 1.0, July 2024

Section	Change Summary
All	Initial release.

www.latticesemi.com