Tri-Speed Ethernet Driver API Reference

Technical Note

FPGA-TN-02341-1.2

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

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
AHB	Advance High Performance
APB	Advanced Peripheral Bus
API	Application Programming Interface
AXI	Advanced extensible Interface
CRC	Cyclic Redundancy Check
FCS	Frame Check Sequence
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
MAC	Media Access Controller
SDK	Software Development Kit
TSE	Tri-Speed Ethernet
TSEMAC	Tri-Speed Ethernet Media Access Controller

1. Introduction

Tri-Speed Ethernet (TSE) IP core is a complex core containing all necessary logic and interfacing and clocking infrastructures necessary to integrate an industry-standard Ethernet PHY with an internal processor efficiently with minimal overhead.

The TSE 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 TSE IP is to ensure that the Media Access rules specified in the 802.3 IEEE standard are met while transmitting a frame of data over Ethernet. On the receiving side, the TSE extracts different components of a frame and transfers them to higher applications through the AXI4-stream interface.

Refer to the Tri-Speed Ethernet IP User Guide (FPGA-IPUG-02084) for more details about the IP core.

1.1. Purpose

TSE and its SDK is a set of application programming interfaces (APIs) that provide access to specific Lattice hardware and software capabilities. 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 CertusPro™-NX and Lattice Avant™ devices. The technical guide assumes readers have expertise in embedded systems and FPGA technologies.

1.3. Driver Version

Driver version: 25.01.00.

1.4. Driver and IP Compatibility

Table 1.1. Driver and IP Version

Driver Version	IP Version
25.01.00	2.0.0

Refer to the Tri-Speed Ethernet IP Release Notes (FPGA-RN-02036) for more information on the driver and IP versions.

Table 1.2. Quick Facts on Driver Test Environment

Hardware Device	Tool Version
CertusPro-NX Versa Board	Lattice Propel™ Builder 2025.1
	Lattice Propel SDK 2025.1
	Lattice Radiant™ software 2025.1
	Radiant Programmer 2025.1

2. API Description

2.1. Ethernet_init()

This API configures the TSEMAC to receive all address frames, sets the MAC address, enables operation at speeds of 10/100/1000 Mbps, and activates both the transmit (TX) and receive (RX) MAC.

```
unsigned char ethernet_init(tsemac_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the following variables that need to be configured before passing it to ethernet_init: • speed_mode—This variable is used to set the speed (10/100/1000 Mbps) for the TSE MAC. • duplex_mode- Represents the Ethernet operation in either half-duplex or full-duplex mode. • adr—Represents the base address of TSEMAC. • mac_upper—Refers to the first four bytes of the MAC address for the TSEMAC. • mac_lower—Refers to the last two bytes of the MAC address for the TSEMAC.	0: failure 1: success

2.2. Ethernet_set_mac_address()

This API sets the six-byte MAC address for a source or destination device for the TSEMAC.

```
unsigned char ethernet_set_mac_address(tsemac_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the following variables that need to be configured before passing it to ethernet_set_mac_address: • adr—Represents the base address of the TSEMAC. • mac_upper—Refers to the first four bytes of the MAC address for the TSEMAC. • mac_lower—Refers to the last two bytes of the MAC address for the TSEMAC.	0: failure 1: success

2.3. Ethernet_get_mac_address()

This API reads the six-byte MAC address from MAC address word 0 and MAC address word 1 register of the TSEMAC, and stores it in mac_upper and mac_lower.

```
unsigned char ethernet_get_mac_address(tsemac_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In/Out	handle	The tsemac_handle_t structure contains the following variables that need to be configured before passing it to ethernet_get_mac_address: • adr—Represents the base address of the TSEMAC.	0: failure 1: success

2.4. Ethernet_set_multicast_address()

This API sets the 8-byte multicast address for multicast frame from the 64-bit hash table. The first four bytes of the 64-bit hash table is stored into Multicast Table Word 0 register and the last four bytes of the 64-bit hash table is stored into Multicast Table Word 1 register.

```
unsigned char ethernet_set_multicast_address(tsemac_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the following variables that need to be configured before passing it to ethernet_set_multicast_address(): • adr—Represents the base address of the TSEMAC. • multicast_upper—Refers to the first four bytes of the multicast value for the TSEMAC. • multicast_lower— Refers to the last four bytes of the multicast address for the TSEMAC.	0: failure 1: success

2.5. Ethernet_get_multicast_address()

This API reads the eight-byte multicast address from Multicast Table Word 0 and Multicast Table Word 1 register of the TSEMAC, and stores it in multicast_upper and multicast_lower.

```
unsigned char ethernet_get_multicast_address(tsemac_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In/Out	handle	The tsemac_handle_t structure contains the following variables that need to be configured before passing it to ethernet_get_mac_address: • adr—Represents the base address of the TSEMAC.	0: failure 1: success

2.6. Ethernet_set_speed()

This API sets the bit for the control register to configure the TSEMAC speed and duplex mode.

```
unsigned char ethernet_set_speed(tsemac_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the following variables that need to be configured before passing it to ethernet_get_multicast_address(): • adr—Represents the base address of the TSEMAC. • speed_mode—Refers to the speed that needs to be configured for the TSEMAC. • duplex_mode – Specifies whether the duplex mode is set to half duplex or full duplex.	0: failure 1: success

2.7. Ethernet_enable_tx_mac()

This API enable TX MAC to transmit frames.

```
unsigned char ethernet_enable_tx_mac(tsemac_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the adr variable, which represents the base address of the TSEMAC. This variable must be configured before it is passed to ethernet_enable_tx_mac().	0: failure 1: success

2.8. Ethernet_enable_rx_mac()

This API enable RX MAC to transmit frames.

```
unsigned char ethernet_enable_rx_mac(tsemac_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the adr variable, which represents the base address of the TSEMAC. This variable must be configured before passing it to ethernet_enable_rx_mac().	0: failure 1: success

2.9. Ethernet_enable_tx_rx_mac()

This API enable TX MAC and RX MAC to transmit and receive frames.

```
unsigned char ethernet_enable_tx_rx_mac(tsemac_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the adr variable, which represents the base address of the TSEMAC. This variable must be configured before passing it to ethernet_enable_tx_rx_mac().	0: failure 1: success

2.10. Ethernet_disable_tx_mac()

This API disable TX MAC to transmit frames.

```
unsigned char ethernet_disable_tx_mac(tsemac_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the adr variable, which represents the base address of the TSEMAC. This variable must be configured before passing it to ethernet_disable_tx_mac().	0: failure 1: success

2.11. Ethernet_disable_rx_mac()

This API disable RX MAC to receive frames.

```
unsigned char ethernet_disable_rx_mac(tsemac_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the adr variable, which represents the base address of the TSEMAC. This variable must be configured before passing it to ethernet_disable_rx_mac().	0: failure 1: success

2.12. Ethernet_disable_tx_rx_mac()

This API disable TX MAC and RX MAC to transmit and receive frames.

```
unsigned char ethernet_disable_tx_rx_mac(tsemac_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the adr variable, which represents the base address of the TSEMAC. This variable must be configured before passing it to ethernet_disable_tx_rx_mac().	0: failure 1: success

2.13. Ethernet_enable_mac_interrupt()

This API enables the MAC interrupt, allowing configuration of 8 available interrupts related to TX and RX FIFO status.

```
unsigned char ethernet_enable_mac_interrupt(tsemac_handle_t *handle, unsigned int
interrupts_enable)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the adr variable, which represents the base address of the TSEMAC. This variable must be configured before passing it to ethernet_enable_mac_interrupt().	0: failure 1: success
In	interrupts_enable	Represents the bit position to enable certain interrupts.	

2.14. Ethernet_disable_mac_interrupt()

This API disables the MAC interrupt and the configuration of 8 available interrupts related to TX and RX FIFO status.

```
unsigned char ethernet_disable_mac_interrupt(tsemac_handle_t *handle, unsigned int
interrupts_disable)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the adr variable, which represents the base address of the TSEMAC. This variable must be configured before passing it to ethernet_disable_mac_interrupt().	0: failure 1: success
In	interrupts_disable	Represents the bit position to disable certain interrupts.	

2.15. Ethernet_get_mac_interrupt_status()

This API retrieves the MAC interrupt status.

```
unsigned char ethernet_get_mac_interrupt_status(tsemac_handle_t *handle, unsigned int *interrupt_status)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the adr variable, which represents the base address of the TSEMAC. This variable must be configured before passing it to ethernet_get_mac_interrupt_status().	0: failure 1: success
Out	interrupt_status	Represents a specific interrupt that has been triggered.	

2.16. Ethernet_clear_mac_interrupt_status()

This API clears the MAC interrupt status, which in turn pulls down the interrupt line.

```
unsigned char ethernet_clear_mac_interrupt_status(tsemac_handle_t *handle, unsigned int interrupt_status)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the adr variable, which represents the base address of the TSEMAC. This variable must be configured before passing it to ethernet_clear_mac_interrupt_status().	0: failure 1: success
In	interrupt_status	Represents a specific interrupt that needs to be cleared.	

2.17. Ethernet_set_tx_almost_full_threshold()

This API sets the transmit (TX) almost full threshold for the Ethernet MAC.

```
unsigned char ethernet_set_tx_almost_full_threshold(tsemac_handle_t *handle, unsigned int tx_fifo_afull_threshold)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the adr variable, which represents the base address of the TSEMAC. This variable must be configured before passing it to ethernet_set_tx_almost_full_threshold().	0: failure 1: success
In	tx_fifo_afull_threshold	Represents the almost full threshold value for TX FIFO. Any value over this threshold triggers an interrupt.	

2.18. Ethernet_set_tx_almost_empty_threshold()

This API sets the transmit (TX) almost empty threshold for the Ethernet MAC.

```
unsigned char ethernet_set_tx_almost_empty_threshold(tsemac_handle_t *handle, unsigned int tx_fifo_aempty_threshold)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the adr variable, which represents the base address of the TSEMAC. This variable must be configured before passing it to ethernet_set_tx_almost_full_threshold().	0: failure 1: success
In	tx_fifo_aempty_threshold	Represents the almost empty threshold value for TX FIFO. Any value over below this threshold triggers an interrupt.	

2.19. Ethernet_set_rx_almost_full_threshold

This API sets the receive (RX) almost full threshold for the Ethernet MAC.

```
unsigned char ethernet_set_rx_almost_full_threshold(tsemac_handle_t *handle, unsigned int rx_fifo_afull_threshold)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the adr variable, which represents the base address of the TSEMAC. This variable must be configured before passing it to ethernet_set_rx_almost_full_threshold().	0: failure 1: success
In	rx_fifo_afull_threshold	Represents the almost full threshold value for RX FIFO. Any value over this threshold triggers an interrupt.	

2.20. Ethernet_set_rx_almost_empty_threshold()

This API sets the RX (receive) FIFO almost empty threshold for the Ethernet MAC.

```
unsigned char ethernet_set_rx_almost_empty_threshold(tsemac_handle_t *handle, unsigned int rx_fifo_aempty_threshold)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the adr variable, which represents the base address of the TSEMAC. This variable must be configured before passing it to ethernet_set_rx_almost_full_threshold().	0: failure 1: success
In	rx_fifo_aempty_threshold	Represents the almost empty threshold value for RX FIFO. Any value below this threshold triggers an interrupt.	

2.21. Ethernet_tx_rx_control_reg_set()

This API reads the transmit and receive status register of the TSEMAC.

```
unsigned char ethernet_tx_rx_control_reg_set(tsemac_handle_t *handle, unsigned int
tx_rx_control_settings)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the adr variable, which represents the base address of the TSEMAC. This variable must be configured before passing it to ethernet_tx_rx_control_reg_set().	0: failure 1: success
In	tx_rx_control_settings	Represents the bit position that the control register sets.	

2.22. Ethernet_mode_reg_read()

This API reads the TSEMAC mode register.

```
unsigned char ethernet_mode_reg_read(tsemac_handle_t *handle, unsigned int *mode_reg)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the adr variable, which represents the base address of the TSEMAC. This variable must be configured before passing it to ethernet_mode_reg_read().	Return the value of the mode register.
Out	mode_reg	Represents the value of mode register.	

2.23. Ethernet_tx_rx_status_reg_read()

This API reads the Ethernet transmit or receive status register.

```
unsigned char ethernet_tx_rx_status_reg_read(tsemac_handle_t *handle, unsigned int
*tx_rx_status)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the adr variable, which represents the base address of the TSEMAC. This variable must be configured before passing it to ethernet_tx_rx_control_reg_set().	0: failure 1: success
Out	tx_rx_status	Represents the value of the transmit and receive status register.	

2.24. Gmii_management_read()

This API reads data from the GMII management interface.

```
unsigned char gmii_management_read(tsemac_handle_t *handle, unsigned char phy_addr, unsigned
char reg_addr, unsigned int *gmii_data)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the adr variable, which represents the base address of the TSEMAC. This variable must be configured before passing it to	0: failure 1: success

In/Out	Parameter	Description	Returns
		gmii_management_read().	
In	phy_addr	Represents the PHY address.	
In	reg_addr	Represents the register address.	
Out	gmii_data	Represents the read value from specific PHY address and register address.	

2.25. Gmii_management_write()

This API writes data to the GMII management interface.

```
unsigned char gmii_management_write(tsemac_handle_t *handle, unsigned char phy_addr, unsigned char reg_addr, unsigned int gmii_data)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the adr variable, which represents the base address of the TSEMAC. This variable must be configured before passing it to gmii_management_read().	0: failure 1: success
In	phy_addr	Represents the PHY address.	
In	reg_addr	Represents the register address.	
In	gmii_data	Represents the write value to specific PHY address and register address.	

2.26. Get_gmii_status()

This API checks the status of the GMII management interface to determine whether it is busy or idle.

```
unsigned char get_gmii_status(tsemac_handle_t *handle, unsigned char *is_gmii_idle)
```

In/Out	Parameter	Description	Returns
In	handle	The tsemac_handle_t structure contains the adr variable, which represents the base address of the TSEMAC. This variable must be configured before passing it to get_gmii_status().	0: failure 1: success
Out	is_gmii_idle	Represents the GMII interface status. Return 1 for idle, return 0 for busy.	

2.27. Ethernet_get_statistic_counter

This API retrieves the value of a specific statistic counter register by adding the statistic counter register offset to the base address. The offset can be found in the Ethernet_reg_name_str(Offset) section. Additionally, the return value of the statistic counter register is in a 64-bit format.

```
static unsigned long long ethernet_get_statstic_counter(unsigned int base_addr, unsigned int offset)
```

In/Out	Parameter	Description	Returns
In	base_addr	Base address of the TSEMAC.	64 bits value of statistic counter value.
In	offset	Represents the offset of the statistic counter register.	

2.28. Ethernet_print_statistic_counter()

This API call ethernet_get_statistic_counter API to retrieve the value of a specific statistic counter register by adding the statistic counter register offset to the base address. It then prints out the value in 16 hexadecimal format along with the corresponding statistic counter name.

```
static void ethernet_print_statistic_counter(unsigned int base_addr, unsigned int offset)
```

In/Out	Parameter	Description	Returns
In	base_addr	Base address of the TSEMAC.	None
In	offset	Represents the offset of the statistic counter register.	

2.29. Ethernet_print_all_statistics_counters()

This API displays the values of all statistics counters registers in 16 hexadecimal format, accompanied by their corresponding names.

```
static void ethernet_print_all_statistics_counters(unsigned int base_addr)
```

In/Out	Parameter	Description	Returns
In	base_addr	Base address of the TSEMAC.	None

3. Function Call Flow Diagrams

3.1. Ethernet_init()

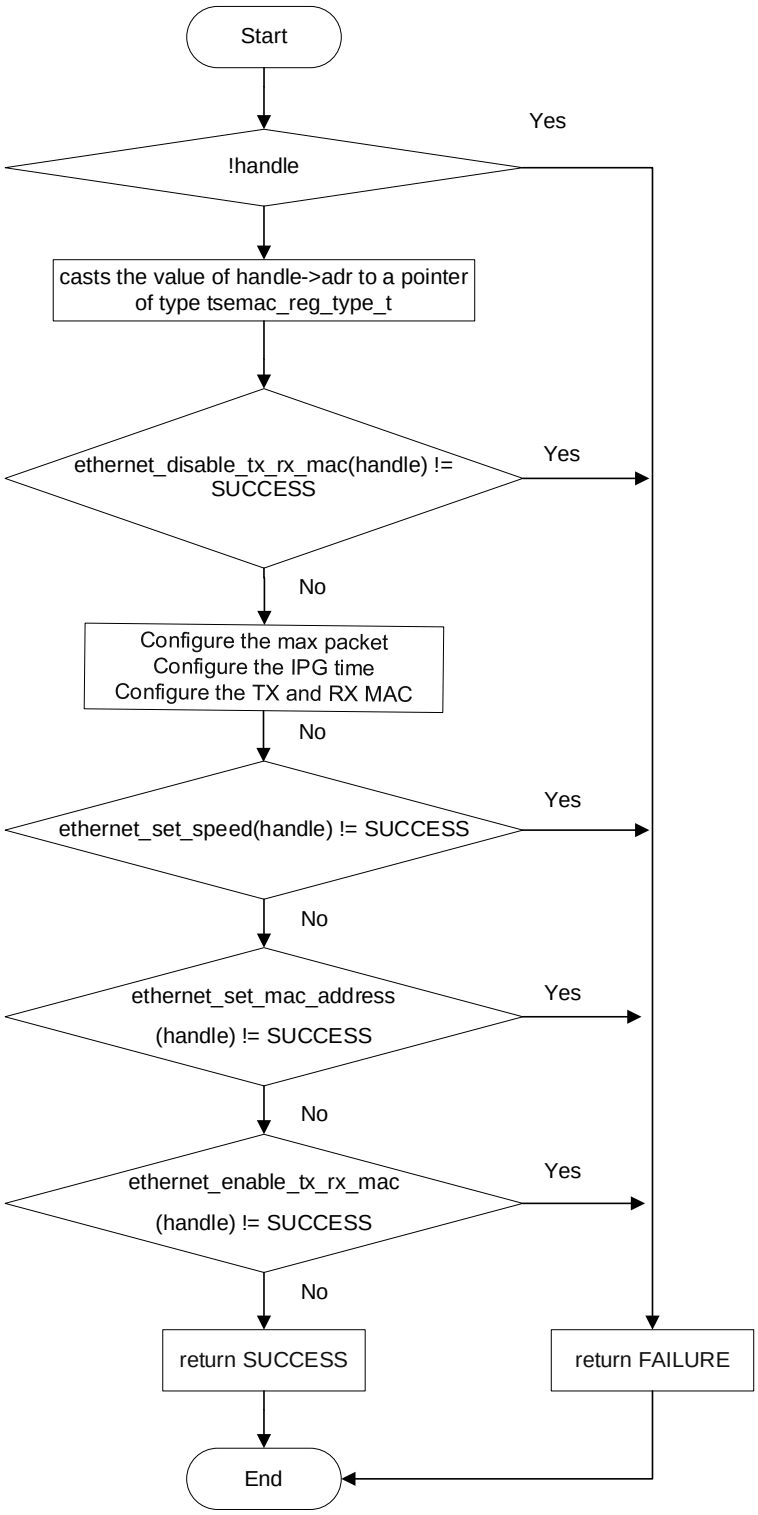

Figure 3.1. ethernet_init()

3.2. Ethernet_set_mac_address()

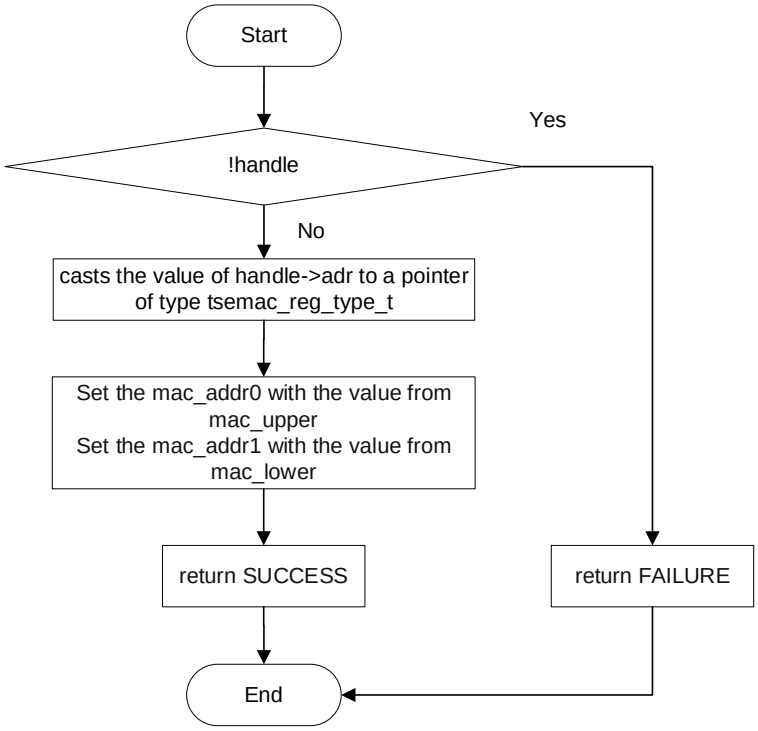

Figure 3.2. ethernet_set_mac_address()

3.3. Ethernet_get_mac_address()

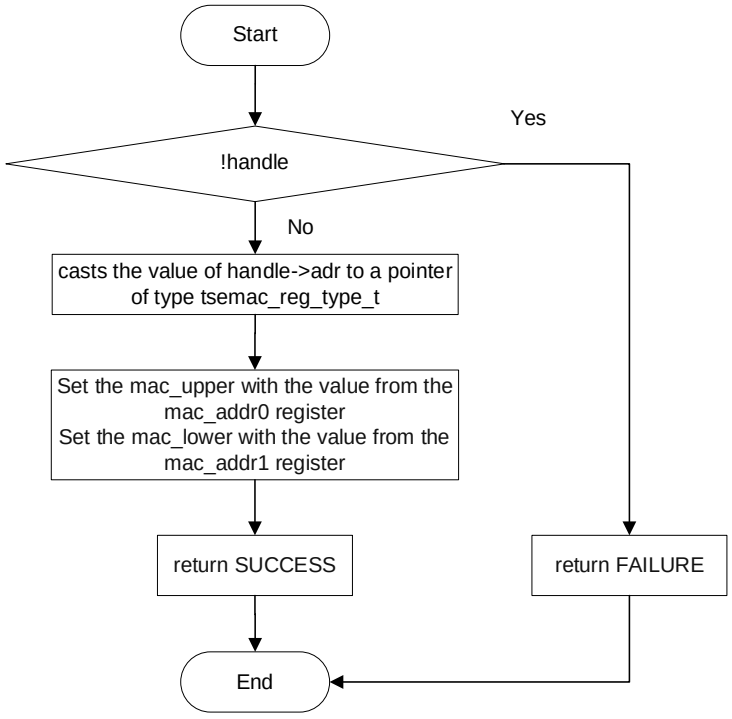

Figure 3.3. ethernet_get_mac_address()

3.4. Ethernet_set_multicast_address()

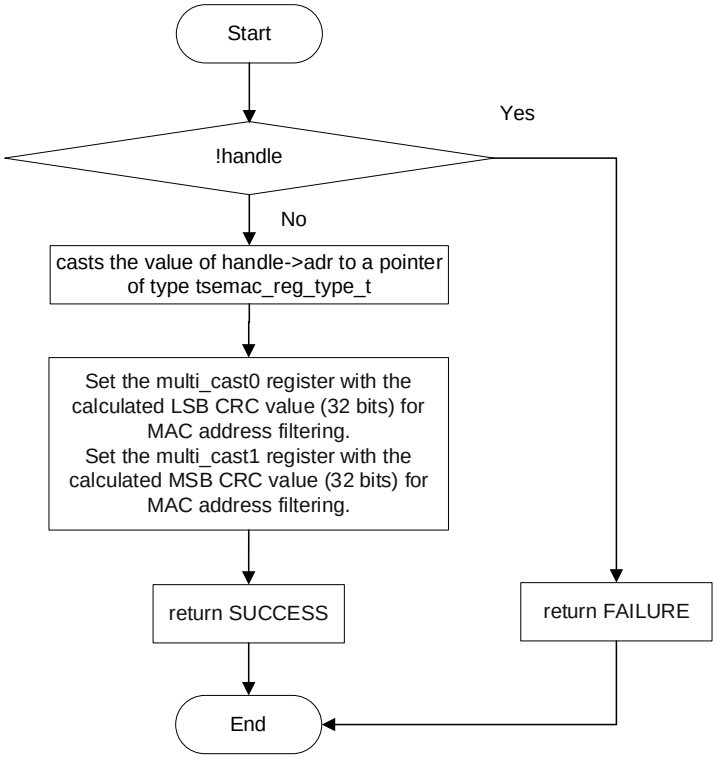

Figure 3.4. ethernet_set_multicast_address()

3.5. Ethernet_get_multicast_address()

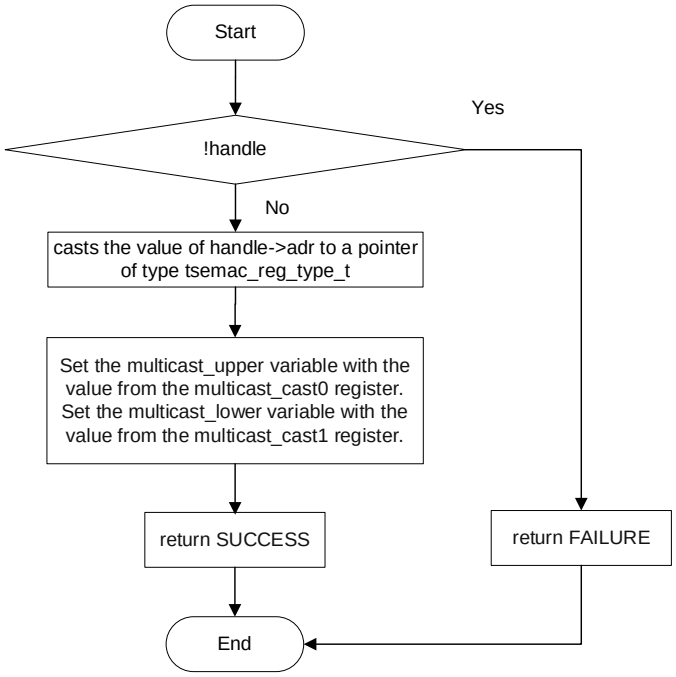

Figure 3.5. ethernet_get_multicast_address()

3.6. Ethernet_set_speed()

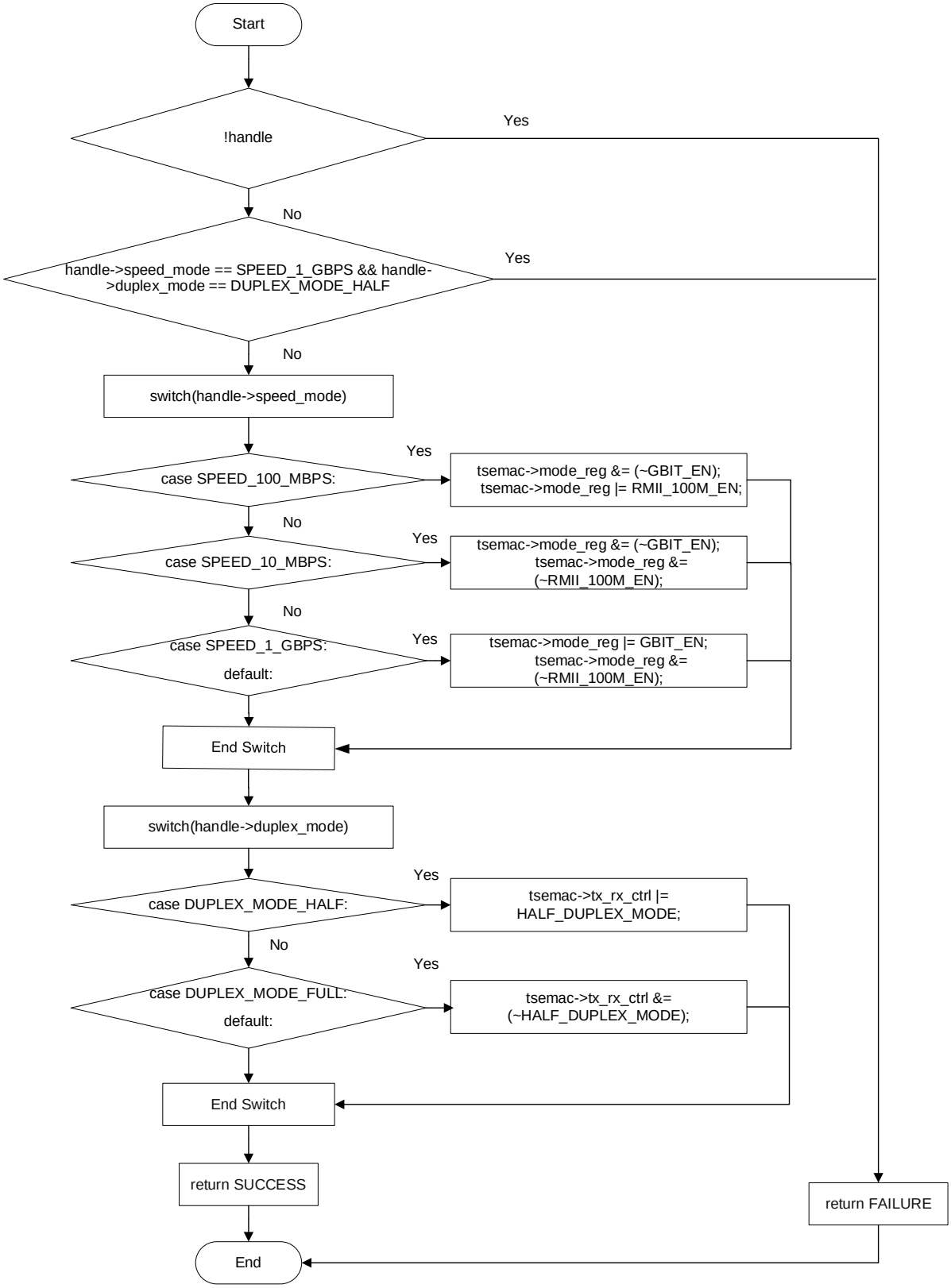

Figure 3.6. ethernet_set_speed()

3.7. Ethernet_enable_tx_mac()

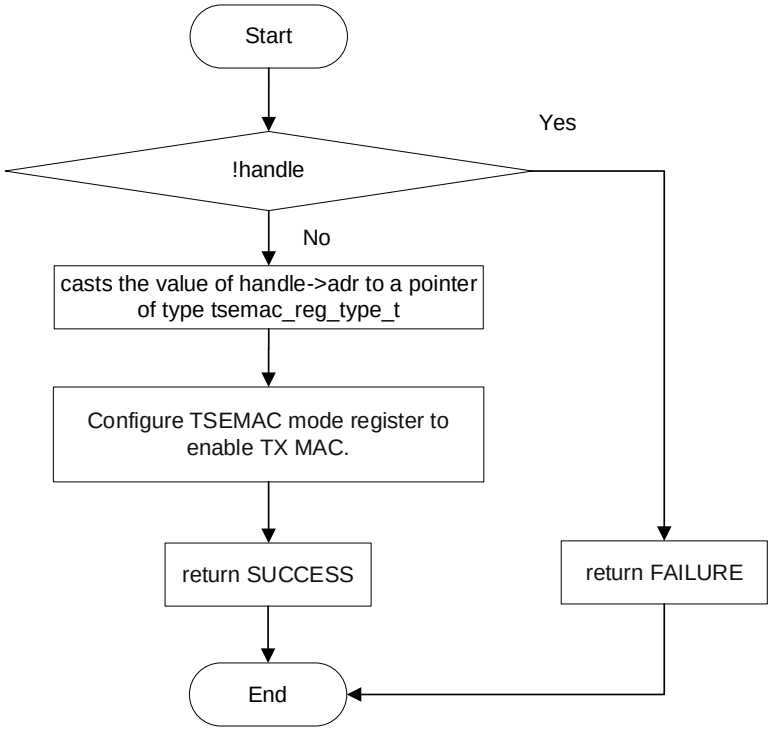

Figure 3.7. ethernet_enable_tx_mac()

3.8. Ethernet_enable_rx_mac()

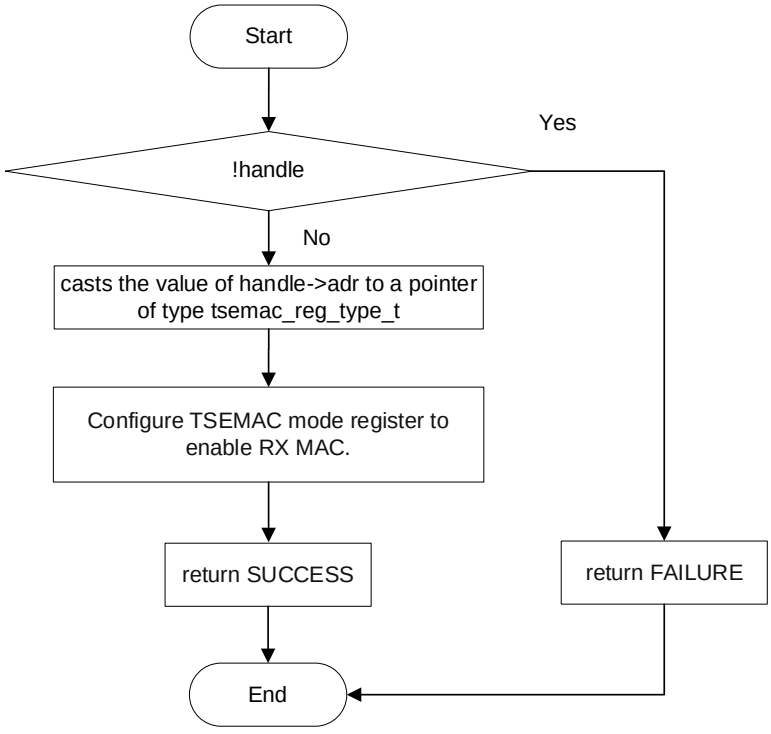

Figure 3.8. ethernet_enable_rx_mac()

3.9. Ethernet_enable_tx_rx_mac()

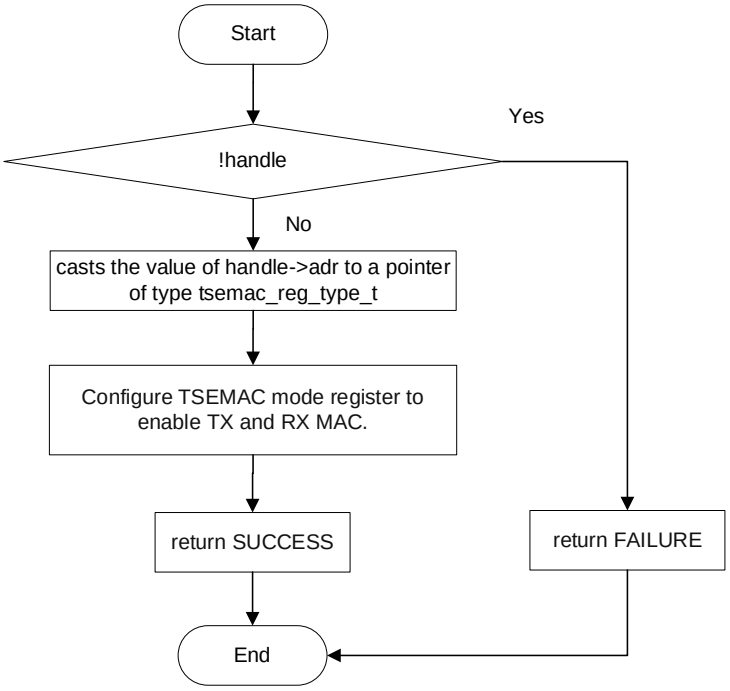

Figure 3.9. ethernet_enable_tx_rx_mac()

3.10. Ethernet_disable_tx_mac()

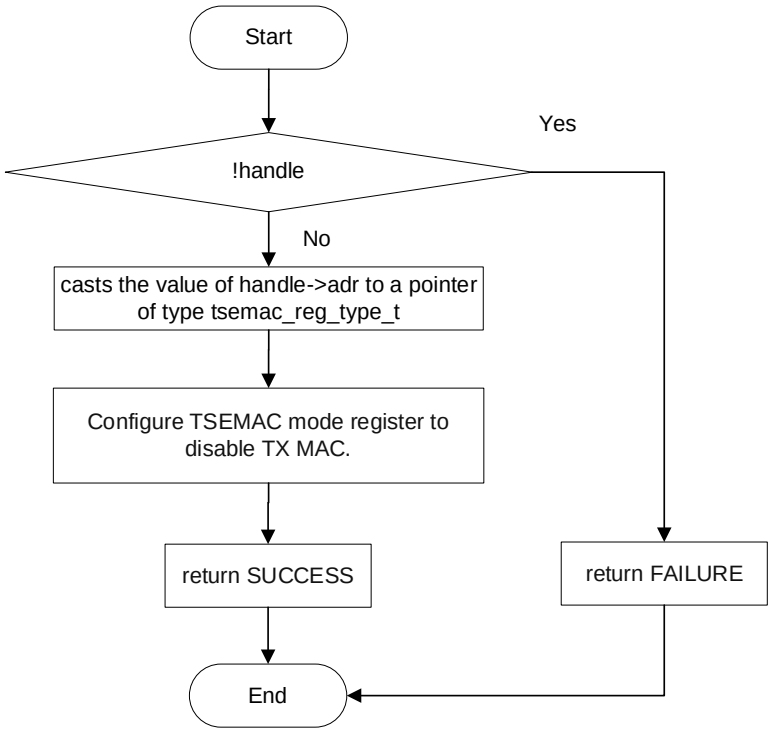

Figure 3.10. ethernet_disable_tx_mac()

3.11. Ethernet_disable_rx_mac()

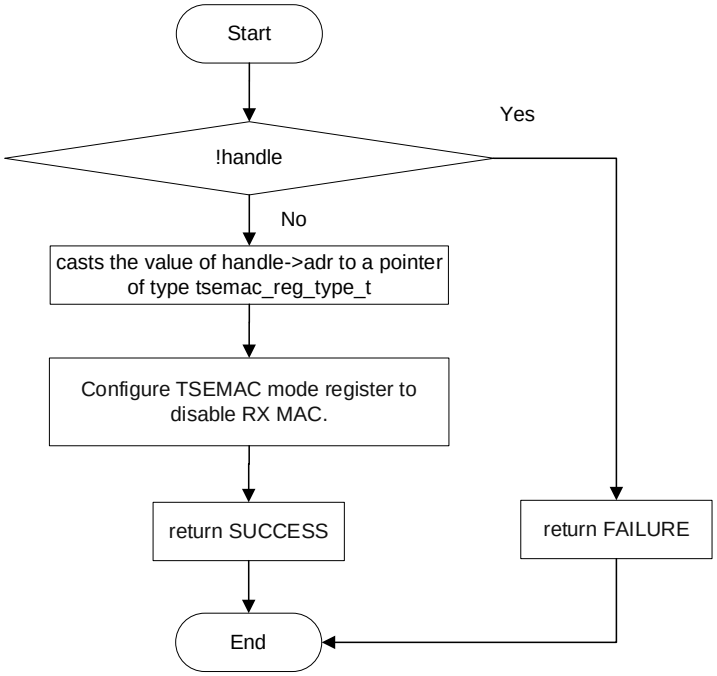

Figure 3.11. ethernet_disable_rx_mac()

3.12. Ethernet_disable_tx_rx_mac()

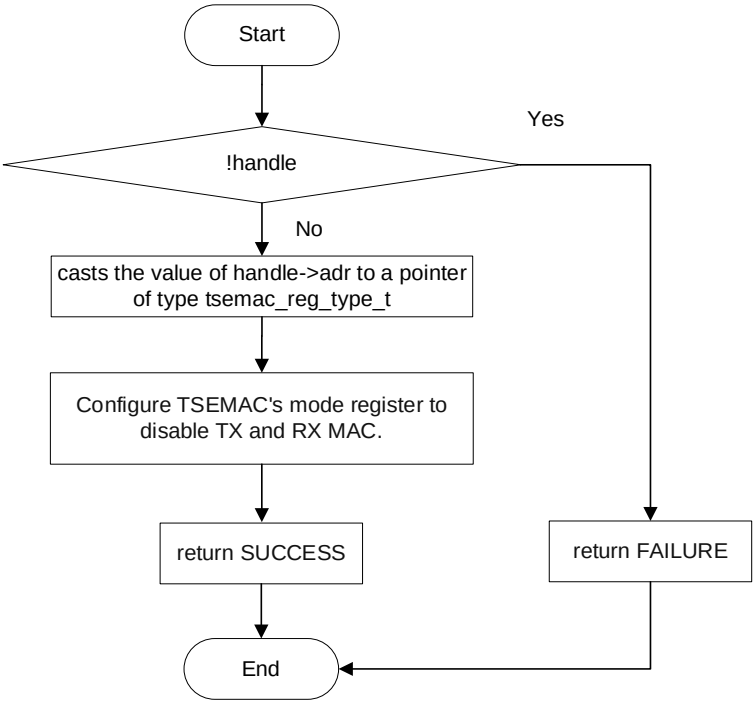

Figure 3.12. ethernet_disable_tx_rx_mac()

3.13. Ethernet_enable_mac_interrupt()

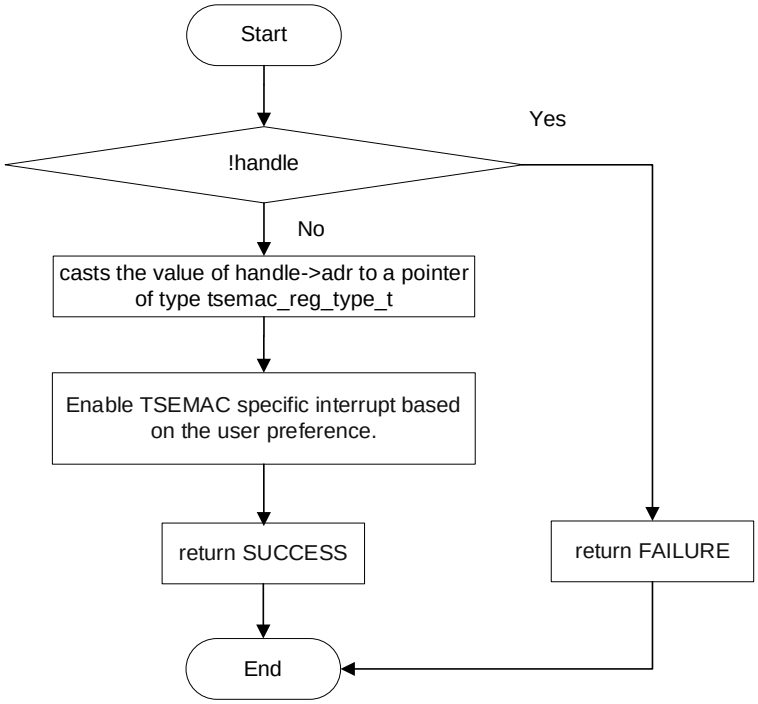

Figure 3.13. ethernet_enable_mac_interrupt()

3.14. Ethernet_disable_mac_interrupt()

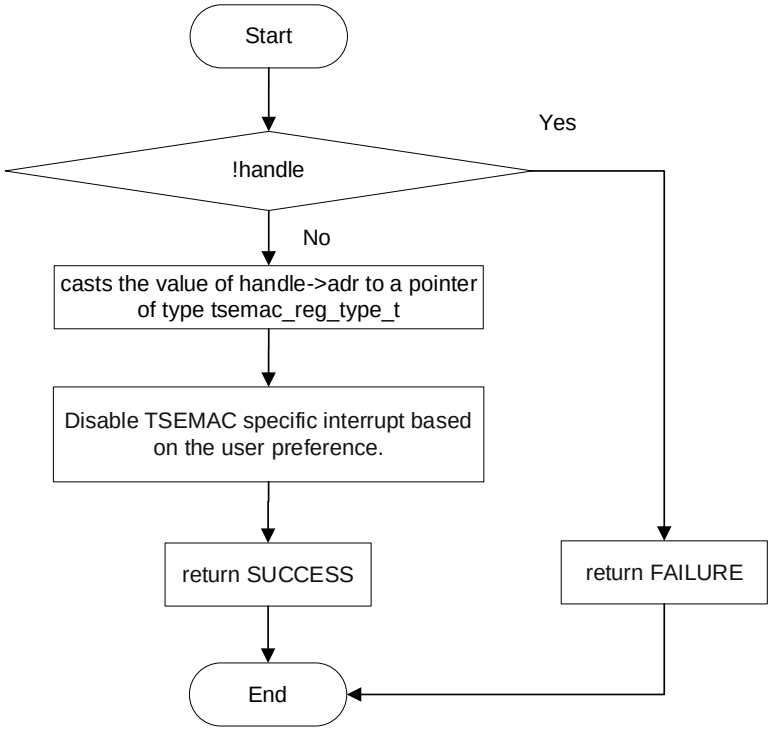

Figure 3.14. ethernet_disable_mac_interrupt()

3.15. Ethernet_get_mac_interrupt_status()

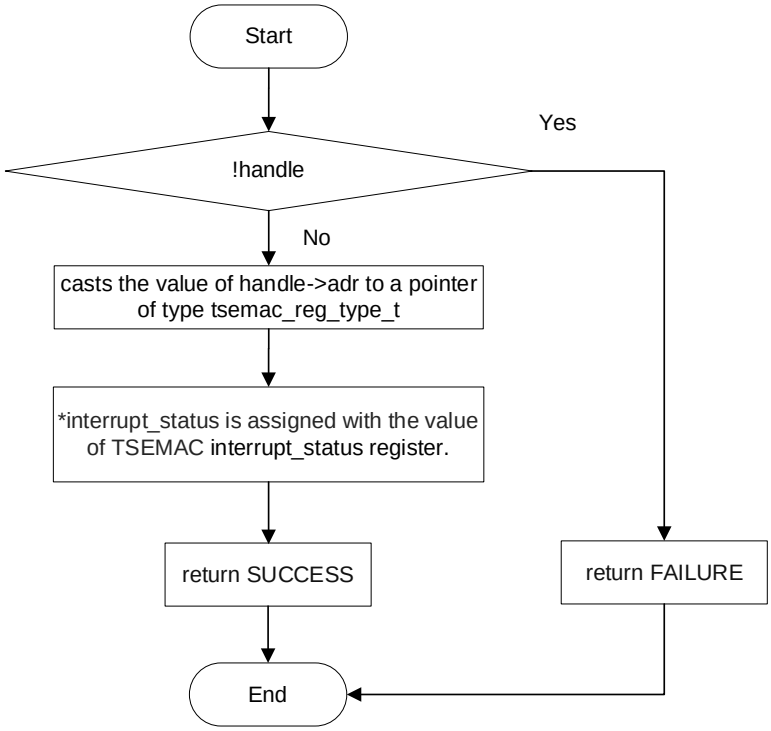

Figure 3.15. ethernet_get_mac_interrupt_status()

3.16. Ethernet_clear_mac_interrupt_status()

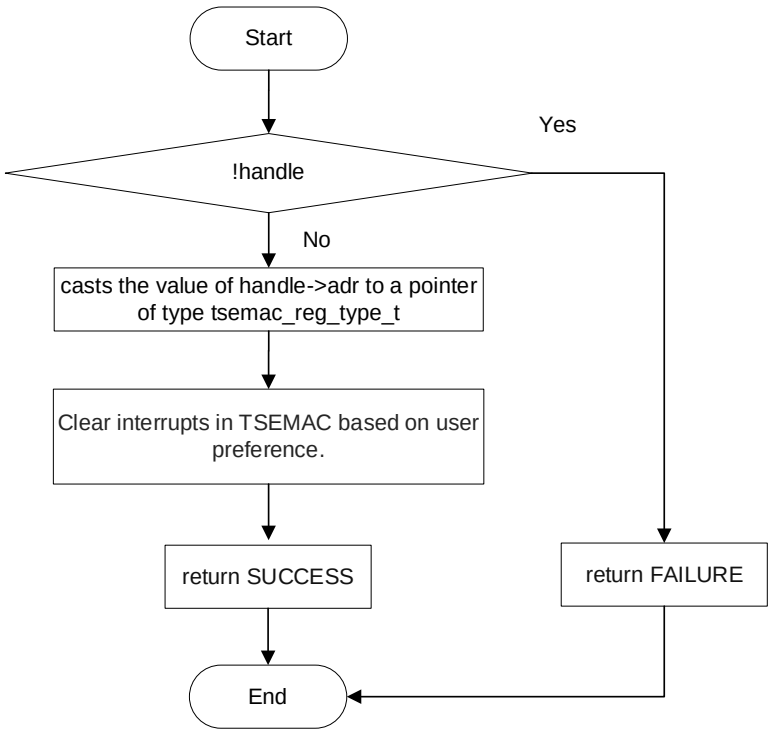

Figure 3.16. ethernet_clear_mac_interrupt_status()

3.17. Ethernet_set_tx_almost_full_threshold()

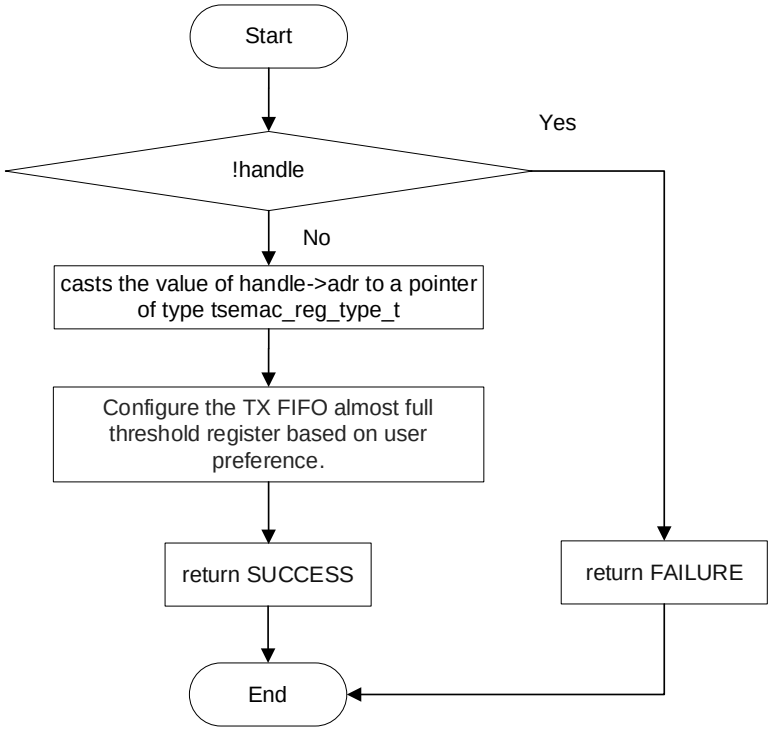

Figure 3.17. ethernet_set_tx_almost_full_threshold()

3.18. Ethernet_set_tx_almost_empty_threshold()

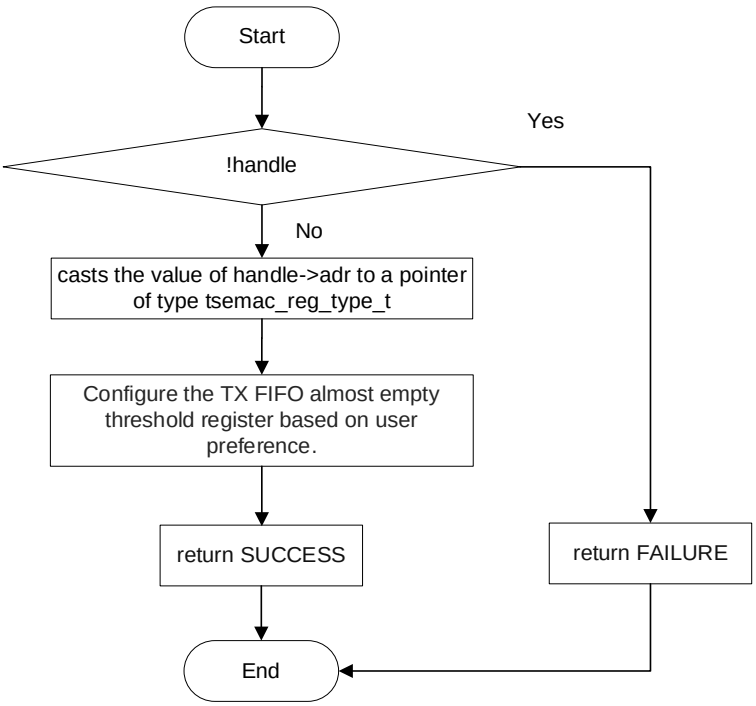

Figure 3.18. ethernet_set_tx_almost_empty_threshold()

3.19. Ethernet_set_rx_almost_full_threshold()

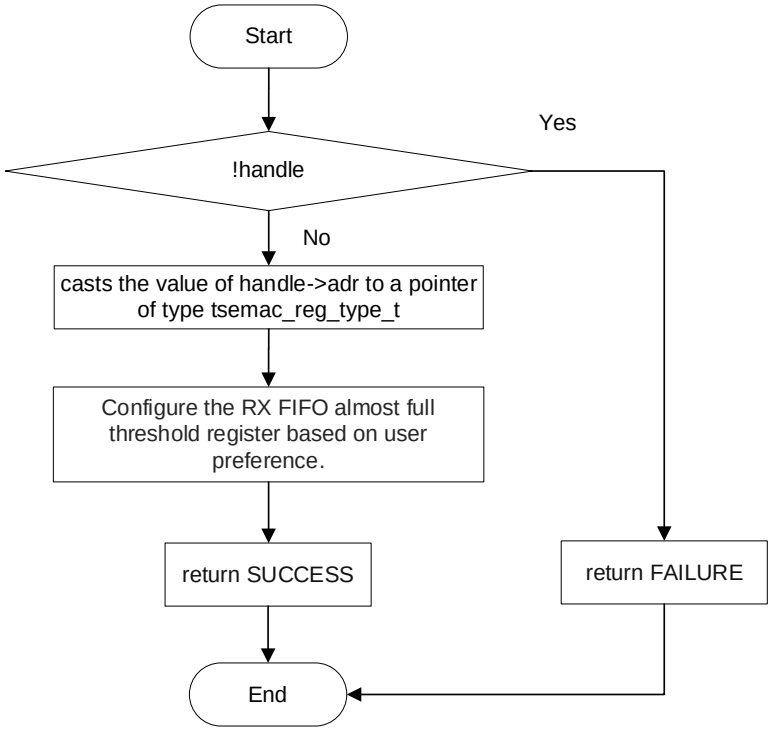

Figure 3.19. ethernet_set_rx_almost_full_threshold()

3.20. Ethernet_set_rx_almost_empty_threshold()

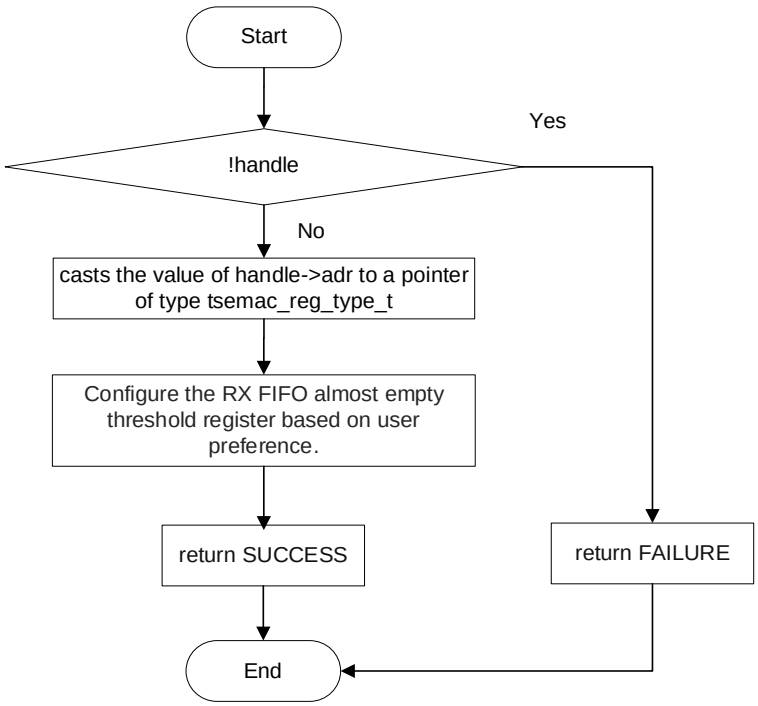

Figure 3.20. ethernet_set_rx_almost_empty_threshold()

3.21. Ethernet_tx_rx_control_reg_set()

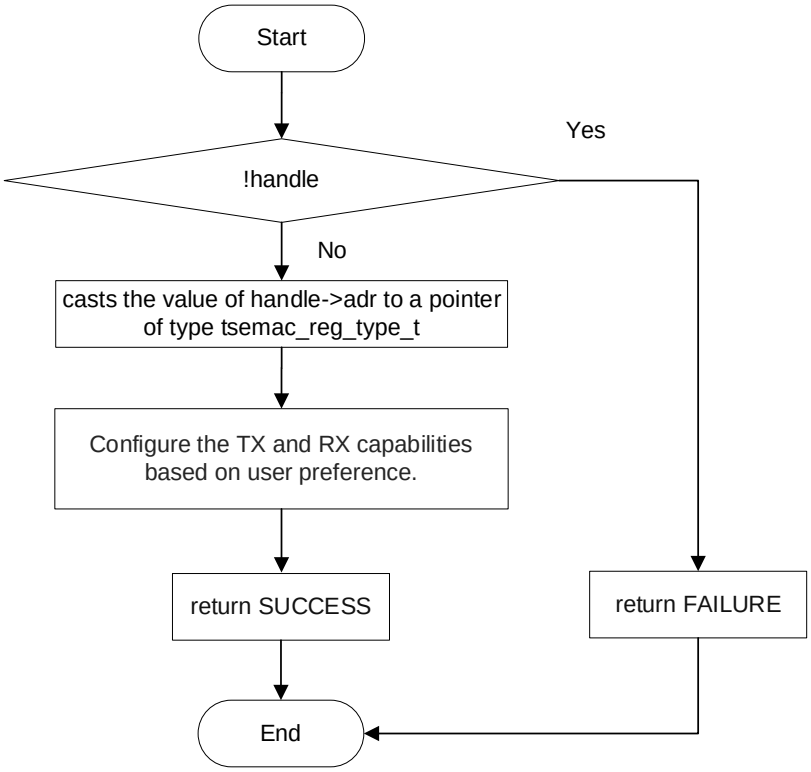

Figure 3.21. ethernet_tx_rx_control_reg_set()

3.22. Ethernet_tx_rx_status_reg_read()

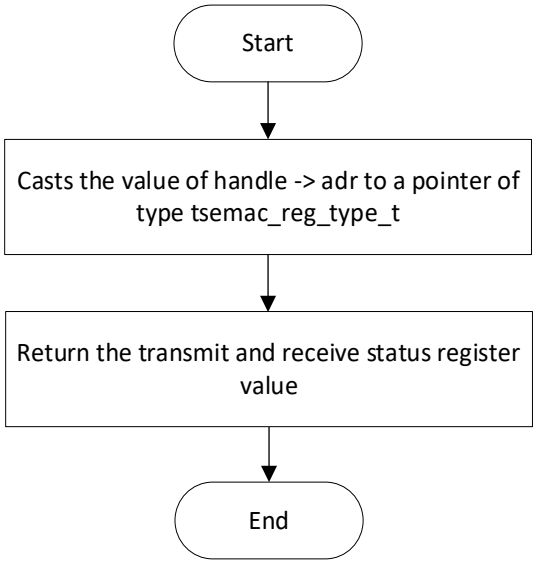

Figure 3.22. ethernet_tx_rx_status_reg_read()

3.23. Ethernet_mode_reg_read()

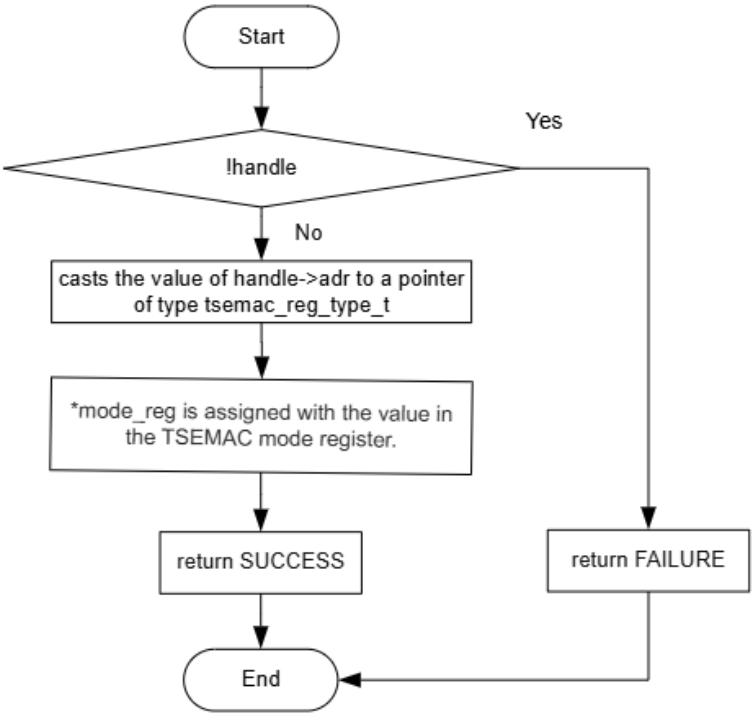

Figure 3.23. ethernet_mode_reg_read()

3.24. get_gmii_status()

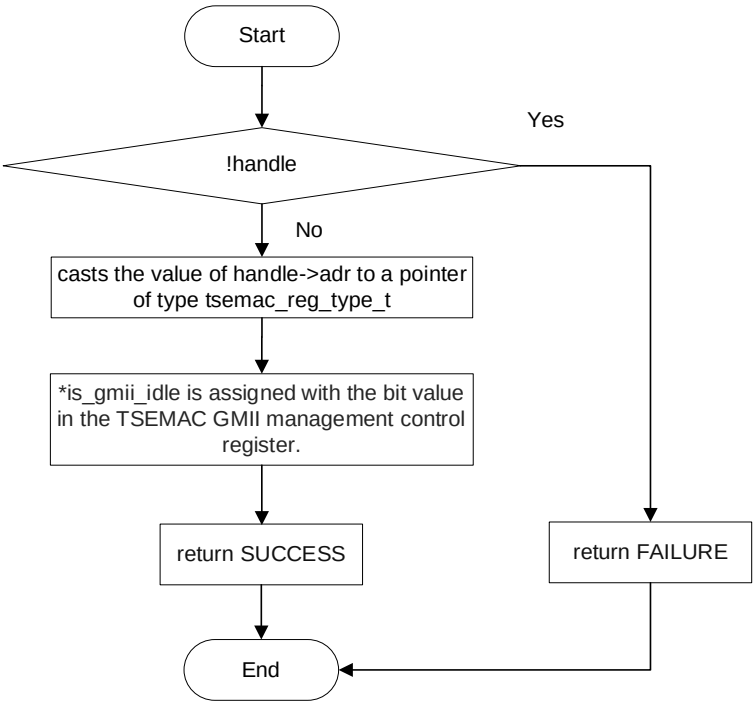

Figure 3.24. get_gmii_status

3.25. gmii_management_read()

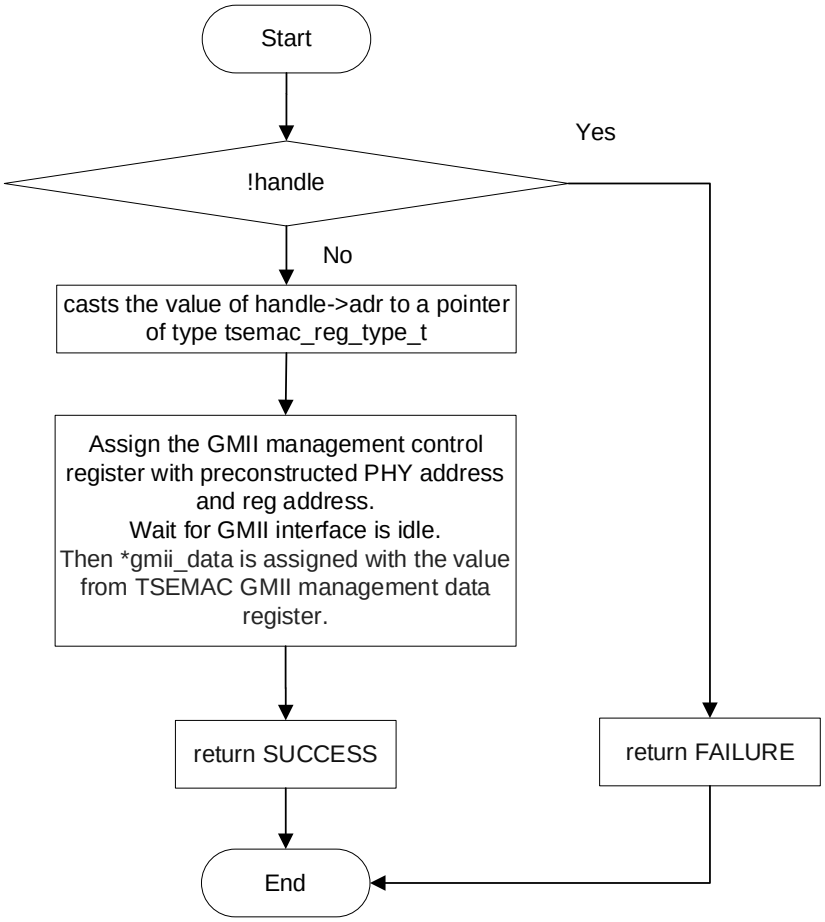

Figure 3.25. gmii_management_read()

3.26. gmii_management_write()

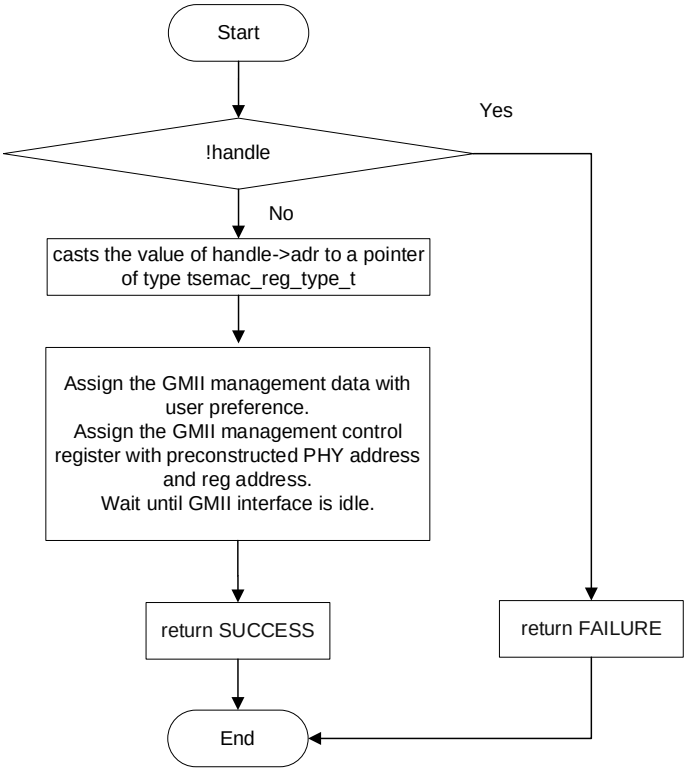

Figure 3.26. gmii_management_write()

3.27. Ethernet_get_statistic_counter()

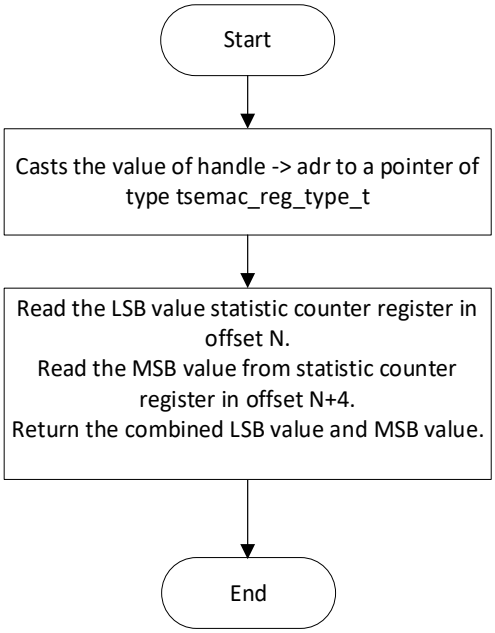

Figure 3.27. ethernet_get_statistic_counter()

3.28. Ethernet_print_statistic_counter()

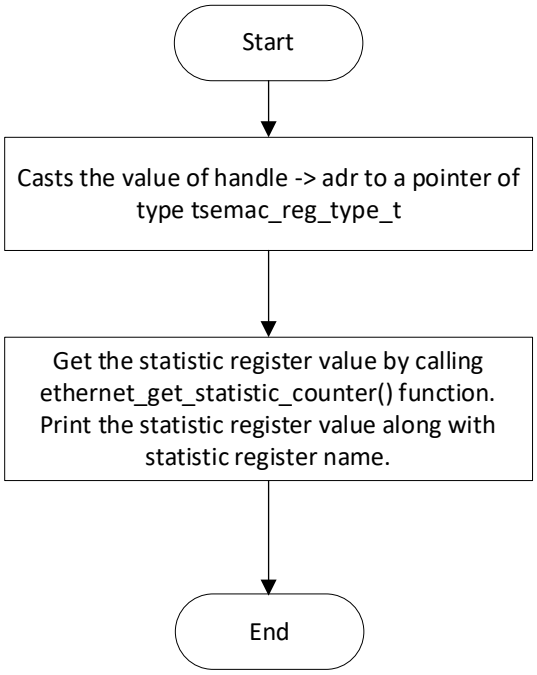

Figure 3.28. ethernet_print_statistic_counter()

3.29. Ethernet_print_all_statistic_counters()

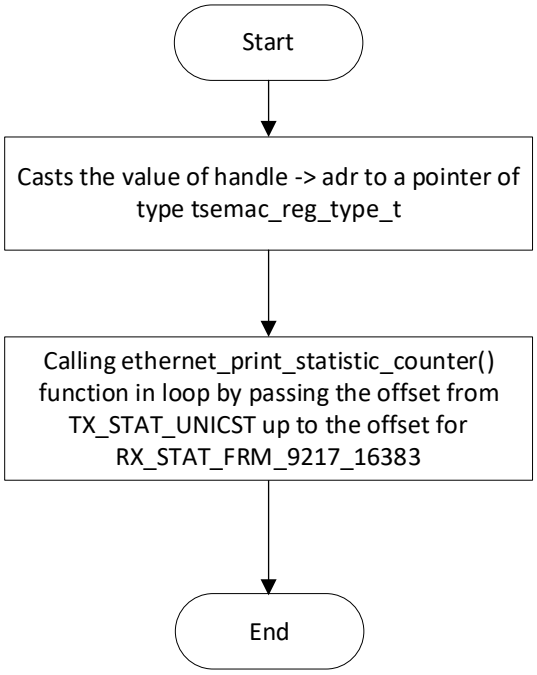

Figure 3.29. ethernet_print_all_statistic_counters()

4. API Data Structures

4.1. tsemac_reg_type_t

This structure assigns offsets for the TSEMAC register.

```
volatile unsigned int mode_reg;          //0x0000
volatile unsigned int tx_rx_ctrl;
volatile unsigned int max_packet_size;
volatile unsigned int ipg;
volatile unsigned int mac_addr0;         //0x0010
volatile unsigned int mac_addr1;
volatile unsigned int tx_rx_status;
volatile unsigned int vlan_tag;
volatile unsigned int gmii_mgmt_ctrl;    //0x0020
volatile unsigned int gmii_mgmt_data;
volatile unsigned int multi_cast0;
volatile unsigned int multi_cast1;
volatile unsigned int pause_opcode;      //0x0030
volatile unsigned int tx_fifo_afull;
volatile unsigned int tx_fifo_aempty;
volatile unsigned int rx_fifo_afull;
volatile unsigned int rx_fifo_aempty;    //0x0040
volatile unsigned int interrupt_status;
volatile unsigned int interrupt_enable;  //0x0048
```

4.2. tsemac_handle_t

This structure is used to declare different types of parameters used for the TSEMAC API. The following table shows the available parameters and description.

Table 4.1. tsemac_handle_t Parameters

Parameter	Description
speed_mode	• Part of the tsemac_handle_t structure. • Used for setting the speed for the TSEMAC.
duplex_mode	• Part of the tsemac_handle_t structure. • Used for setting the half duplex or full duplex mode for the TSEMAC.
adr	• Part of the tsemac_handle_t structure. • Represents the base address of the TSEMAC.
frame_length	• Part of the tsemac_handle_t structure. • Represents the length of the Ethernet packet that is transmitted to the TSEMAC.
mac_upper	• Part of the tsemac_handle_t structure. • Represents the variable setting to obtain the first four bytes of the MAC address for the TSEMAC.
mac_lower	• Part of the tsemac_handle_t structure. • Represents the variable setting to obtain the last two bytes of the MAC address for the TSEMAC.
multicast_upper	• Part of the tsemac_handle_t structure. • Represents the variable setting to obtain the first four bytes of the 64-bit hash for the TSEMAC.
multicast_lower	• Part of the tsemac_handle_t structure. • Represents the variable setting to obtain the last four bytes of the 64-bit hash for the TSEMAC.
tx_rx_ctrl_var	• Part of the tsemac_handle_t structure. • Represents the transmit and receive control register configuration for the TSEMAC.

5. API Variables

The following table shows the variables and their description.

Table 5.1. Variable Description

Variable	Description
handle	• Used in the TSEMAC API. • A structure variable that consists of different variables.
status	Returns success or failure from the API.

6. API Macros

6.1. Success and Failure

```
#define     SUCCESS                 1
#define     FAILURE                 0
```

6.2. IPG Time

```
#define     IPG_TIME                12
```

6.3. Maximum Packet Size

```
#define     MAX_PACKET_SIZE         1500
```

The MAX_PACKET_SIZE is user dependent. You can set this API for the TSEMAC as per requirement. A frame that transmits more than the maximum packet size is known as a long frame.

6.4. TSE MAC Speed Mode

```
#define     SPEED_10_MBPS           1
#define     SPEED_100_MBPS          2
#define     SPEED_1_GBPS            3
```

6.5. TSE MAC Duplex Mode

```
#define     DUPLEX_MODE_HALF        4
#define     DUPLEX_MODE_FULL        5
```

6.6. Mode Register Macros

```
#define     RMII_100M_EN            (1U << 4)
#define     TX_EN                   (1U << 3)
#define     RX_EN                   (1U << 2)
#define     FC_EN                   (1U << 1)
#define     GBIT_EN                 (1U << 0)
```

6.7. Transmit and Receive Control Register Macros

```
#define     RCV_SHORT_FRAMES        (1U << 8)
#define     RCV_BROADCAST           (1U << 7)
#define     DROP_CONTROL            (1U << 6)
#define     HALF_DUPLEX_MODE        (1U << 5)
#define     RCV_MULTICAST           (1U << 4)
#define     RCV_PAUSE_FRAME         (1U << 3)
#define     TX_DISCARD_FCS          (1U << 2)
#define     RCV_DISCARD_FCS         (1U << 1)
#define     RCV_ALL_ADDR_FRAME      (1U << 0)
```

6.8. Transmit and Receive Status Register Macros

```
#define      Rx_MAC_IDLE              (1U << 10)
#define      TAGGED_FRAME              (1U << 9)
#define      BROADCAST_FRAME           (1U << 8)
#define      MULTICAST_CAST            (1U << 7)
#define      IPG_SHRINK                (1U << 6)
#define      SHORT_FRAME               (1U << 5)
#define      LONG_FRAME                (1U << 4)
#define      FRAME_ERROR               (1U << 3)
#define      CRC_ERROR                 (1U << 2)
#define      PAUSE_FRAME               (1U << 1)
#define      Tx_MAC_IDLE               (1U << 0)
```

6.9. GMII Management Register Access Control Register Macros

```
#define      CMD_FINISHED              (1UL << 14)
#define      RW_PHYREG_POS             13
#define      R_PHYREG_POS              0
#define      W_PHYREG_POS              1
#define      REG_START_POS             0
#define      PHY_START_POS             8
#define      MAX_REG_ADDR              31
#define      MAX_PHY_ADDR              31
```

6.10. Interrupt Status Register Macros

```
#define      TX_OVERFLOW_INT           (1U << 7)
#define      TX_FULL_INT               (1U << 6)
#define      TX_AFULL_INT              (1U << 5)
#define      TX_AEMPTY_INT             (1U << 4)
#define      RX_UNDERFLOW_INT          (1U << 3)
#define      RX_EMPTY_INT              (1U << 2)
#define      RX_AEMPTY_INT             (1U << 1)
#define      RX_AFULL_INT              (1U << 0)
```

6.11. Interrupt Enable Register Macros

```
#define      TX_OVERFLOW_EN            (1U << 7)
#define      TX_FULL_EN                (1U << 6)
#define      TX_AFULL_EN               (1U << 5)
#define      TX_AEMPTY_EN              (1U << 4)
#define      RX_UNDERFLOW_EN           (1U << 3)
#define      RX_EMPTY_EN               (1U << 2)
#define      RX_AEMPTY_EN              (1U << 1)
#define      RX_AFULL_EN               (1U << 0)
```

6.12. Statistics Counters Registers Offset

```
#define TX_STAT_UNICST                          0x04C
#define TX_STAT_MULTCST                         0x054
#define TX_STAT_BRDCST                          0x05C
#define TX_STAT_BADFCS                          0x064
#define TX_STAT_JMBO                            0x06C
#define TX_STAT_UNDER_RUN                       0x074
#define TX_STAT_PAUSE                           0x07C
#define TX_STAT_VLN_TG                          0x084
#define TX_STAT_FRM_LNGTH                       0x08C
#define TX_STAT_DEFERRED_TRANS                  0x094
#define TX_STAT_EXCESSIVE_DEFERRED_TRANS        0x09C
#define TX_STAT_LATE_COL                        0x0A4
#define TX_STAT_EXCESSIVE_COL                   0x0AC
#define TX_STAT_NUM_EARLY_COL                   0x0B4
#define TX_STAT_SHRT_FRM_DIS_FCS                0x0BC
#define TX_STAT_PTP1588_FRM                     0x0C4
#define TX_STAT_FRM_64                          0x0CC
#define TX_STAT_FRM_65_127                      0x0D4
#define TX_STAT_FRM_128_255                     0x0DC
#define TX_STAT_FRM_256_511                     0x0E4
#define TX_STAT_FRM_512_1023                    0x0EC
#define TX_STAT_FRM_1024_1518                   0x0F4
#define TX_STAT_FRM_1519_2047                   0x0FC
#define TX_STAT_FRM_2048_4095                   0x104
#define TX_STAT_FRM_4096_9216                   0x10C
#define TX_STAT_FRM_9217_16383                  0x114
#define RX_STAT_FRM_LNGTH                       0x11C
#define RX_STAT_VLN_TG                          0x124
#define RX_STAT_PAUSE                           0x12C
#define RX_STAT_CTRL                            0x134
#define RX_STAT_UNSP_OPCODE                     0x13C
#define RX_STAT_DRIBB_NIBB                      0x144
#define RX_STAT_BRDCST                          0x14C
#define RX_STAT_MULTCST                         0x154
#define RX_STAT_UNICST                          0x15C
#define RX_STAT_RCVD_OK                         0x164
#define RX_STAT_LNGTH_ERR                       0x16C
#define RX_STAT_CRC_ERR                         0x174
#define RX_STAT_PKT_IGNORE                      0x17C
#define RX_STAT_PREVIOUS_CARRIER_EVENT          0x184
#define RX_STAT_PTP1588_FRM                     0x18C
#define RX_STAT_IPG_VIOL                        0x194
#define RX_STAT_SHRT_FRM                        0x19C
#define RX_STAT_LNG_FRM                         0x1A4
#define RX_STAT_FRM_UNDERSIZE                   0x1AC
#define RX_STAT_FRM_FRAGMENTS                   0x1B4
#define RX_STAT_FRM_JABBER                      0x1BC
#define RX_STAT_FRM_64_GOOD_CRC                 0x1C4
#define RX_STAT_FRM_1518_GOOD_CRC               0x1CC
#define RX_STAT_FRM_64                          0x1D4
```

```
#define RX_STAT_FRM_65_127                  0x1DC
#define RX_STAT_FRM_128_255                 0x1E4
#define RX_STAT_FRM_256_511                 0x1EC
#define RX_STAT_FRM_512_1023                0x1F4
#define RX_STAT_FRM_1024_1518               0x1FC
#define RX_STAT_FRM_1519_2047               0x204
#define RX_STAT_FRM_2048_4095               0x20C
#define RX_STAT_FRM_4096_9216               0x214
#define RX_STAT_FRM_9217_16383              0x21C
```

6.13. Ethernet_reg_name_str(Offset)

```
#define ethernet_reg_name_str(Offset)
```

The ethernet_reg_name_str(Offset) macro retrieves the names of statistics counters registers as strings, taking the offset of the statistics counters registers as input.

References

- Tri-Speed Ethernet IP Core User Guide (FPGA-IPUG-02084)
- Tri-Speed Ethernet IP Release Notes (FPGA-RN-02036)
- Avant-E web page
- Avant-G web page
- Avant-X web page
- Certus-NX web page
- CertusPro-NX web page
- CrossLink-NX web page
- Mach-NX web page
- MachXO2 web page
- MachXO3D web page
- MachXO5-NX web page
- Lattice Solutions IP Cores web page
- Lattice Radiant Software FPGA 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, June 2025

Section	Change Summary
All	• Renamed document title *TSE MAC Driver API Reference* to *Tri-Speed Ethernet Driver API Reference*. • Renamed *Tri-Speed Ethernet MAC* to *Tri-Speed Ethernet*.
Introduction	• Updated the Driver and IP Compatibility section. • Renamed *TSEMAC IP* to *TSE IP* in this section.

API Description	• Updated the following API description:
	  • Ethernet_init() section.
	  • Ethernet_get_mac_address() section.
	  • Ethernet_get_multicast_address() section.
	  • Ethernet_set_speed() section.
	  • Ethernet_mode_reg_read() section.
	  • Ethernet_tx_rx_status_reg_read() section.
	• Added the following API description:
	  • Ethernet_enable_tx_mac() section.
	  • Ethernet_enable_rx_mac() section.
	  • Ethernet_disable_tx_mac() section.
	  • Ethernet_disable_rx_mac() section.
	  • Ethernet_enable_mac_interrupt() section.
	  • Ethernet_disable_mac_interrupt() section.
	  • Ethernet_get_mac_interrupt_status() section.
	  • Ethernet_clear_mac_interrupt_status() section.
	  • Ethernet_set_tx_almost_full_threshold() section.
	  • Ethernet_set_tx_almost_empty_threshold() section.
	  • Ethernet_set_rx_almost_full_threshold section.
	  • Ethernet_set_rx_almost_empty_threshold() section.
	  • Ethernet_tx_rx_control_reg_set() section.
	  • Gmii_management_read() section.
	  • Gmii_management_write() section.
	  • Get_gmii_status() section.

Function Call Flow Diagrams	• Updated the following flow diagrams:
	  • Figure 3.1. ethernet_init().
	  • Figure 3.2. ethernet_set_mac_address().
	  • Figure 3.3. ethernet_get_mac_address().
	  • Figure 3.4. ethernet_set_multicast_address().
	  • Figure 3.5. ethernet_get_multicast_address().
	  • Figure 3.6. ethernet_set_speed().
	  • Figure 3.9. ethernet_enable_tx_rx_mac().
	  • Figure 3.12. ethernet_disable_tx_rx_mac().
	  • Figure 3.21. ethernet_tx_rx_control_reg_set().
	  • Figure 3.23. ethernet_mode_reg_read().
	• Added the following flow diagrams:
	  • Figure 3.7. ethernet_enable_tx_mac().
	  • Figure 3.8. ethernet_enable_rx_mac().
	  • Figure 3.10. ethernet_disable_tx_mac().
	  • Figure 3.11. ethernet_disable_rx_mac().
	  • Figure 3.13. ethernet_enable_mac_interrupt().
	  • Figure 3.14. ethernet_disable_mac_interrupt().
	  • Figure 3.15. ethernet_get_mac_interrupt_status().

Section	Change Summary
	• Figure 3.16. ethernet_clear_mac_interrupt_status(). • Figure 3.17. ethernet_set_tx_almost_full_threshold(). • Figure 3.18. ethernet_set_tx_almost_empty_threshold(). • Figure 3.19. ethernet_set_rx_almost_full_threshold(). • Figure 3.20. ethernet_set_rx_almost_empty_threshold(). • Figure 3.24. get_gmii_status. • Figure 3.25. gmii_management_read(). • Figure 3.26. gmii_management_write().

API Data Structures	• Removed the following parameters from Table 4.1. tsemac_handle_t Parameters:
	  • enable_tx_mac.
	  • enable_rx_mac.
	• Added the duplex mode parameter to Table 4.1. tsemac_handle_t Parameters.

API Macros	• Added the following sections:
	  • TSE MAC Speed Mode section.
	  • TSE MAC Duplex Mode section.
	  • Mode Register Macros section.
	  • Transmit and Receive Status Register Macros section.
	  • GMII Management Register Access Control Register Macros section.
	  • Interrupt Status Register Macros section.
	  • Interrupt Enable Register Macros section.
	• Updated the Transmit and Receive Control Register Macros section.

References	Added the following reference: *Tri-Speed Ethernet IP Release Notes (FPGA-RN-02036).*

Revision 1.1, July 2024

Section	Change Summary
Abbreviations in This Document	Added the FCS abbreviation.
Introduction	• Updated the Audience section. • Updated the Driver Version section. • Updated Table 1.1. Driver and IP Version and Table 1.2. Quick Facts on Driver Test Environment in the Driver and IP Compatibility section.

API Description	• Updated the following subsections:
	  • Ethernet_init() section.
	  • Ethernet_packet_handle() section.
	  • Ethernet_set_mac_address() section.
	  • Ethernet_get_mac_address() section.
	  • Ethernet_set_multicast_address() section.
	  • Ethernet_get_multicast_address() section.
	  • Ethernet_tx_rx_status_reg_read() section.
	  • Ethernet_mode_reg_read() section.
	  • Ethernet_tx_rx_control_reg_set() section.
	  • Ethernet_set_speed() section.
	• Added the following new subsections:
	  • Ethernet_enable_tx_rx_mac() section.
	  • Ethernet_disable_tx_rx_mac() section.
	  • Ethernet_get_statistic_counter section.
	  • Ethernet_print_statistic_counter() section.
	  • Ethernet_print_all_statistics_counters() section.

Function Call Flow	• Updated the following figures:

Section	Change Summary
Diagrams	• Figure 3.1. ethernet_init(). • Figure 3.2. ethernet_packet_handle(). • Figure 3.3. ethernet_set_mac_address(). • Figure 3.4. ethernet_get_mac_address(). • Figure 3.5. ethernet_set_multicast_address(). • Figure 3.6. ethernet_get_multicast_address(). • Figure 3.7. ethernet_set_speed(). • Added the following new figures: • Figure 3.8. ethernet_enable_tx_rx_mac(). • Figure 3.9. ethernet_disable_tx_rx_mac(). • Figure 3.13. ethernet_get_statistic_counter(). • Figure 3.14. ethernet_print_statistic_counter(). • Figure 3.15. ethernet_print_all_statistic_counters().
API Data Structures	• Updated the tsemac_reg_type_t section. • Updated Table 4.1. tsemac_handle_t Parameters.
API Enum	• Updated the Speed_mode_set section. • Removed the statistics_counter_reg section.
API Macros	• Removed the following sections: • Frame Length section. • Shift Value section. • Index Value section. • Updated the Maximum Packet size section. • Added the following sections: • Statistics Counters Registers Offset. • Ethernet_reg_name_str(Offset).

Revision 1.0, December 2023

Section	Change Summary
All	Initial release.

www.latticesemi.com