TSE MAC Driver API Reference

Technical Note

FPGA-TN-02341-1.1

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.

Contents

Contents....3
Abbreviations in This Document....7
1. Introduction....8
1.1. Purpose....8
1.2. Audience....8
1.3. Driver Version....8
1.4. Driver and IP Compatibility....8
2. API Description....9
2.1. Ethernet_init()....9
2.2. Ethernet_packet_handle()....9
2.3. Ethernet_set_mac_address()....10
2.4. Ethernet_get_mac_address()....10
2.5. Ethernet_set_multicast_address()....10
2.6. Ethernet_get_multicast_address()....11
2.7. Ethernet_set_speed()....11
2.8. Ethernet_enable_tx_rx_mac()....11
2.9. Ethernet_disable_tx_rx_mac()....11
2.10. Ethernet_tx_rx_status_reg_read()....12
2.11. Ethernet_mode_reg_read()....12
2.12. Ethernet_tx_rx_control_reg_set()....12
2.13. Ethernet_get_statistic_counter....12
2.14. Ethernet_print_statistic_counter()....13
2.15. Ethernet_print_all_statistics_counters()....13
3. Function Call Flow Diagrams....14
3.1. Ethernet_init()....14
3.2. Ethernet_packet_handle()....14
3.3. Ethernet_set_mac_address()....15
3.4. Ethernet_get_mac_address()....15
3.5. Ethernet_set_multicast_address()....16
3.6. Ethernet_get_multicast_address()....16
3.7. Ethernet_set_speed()....17
3.8. Ethernet_enable_tx_rx_mac()....18
3.9. Ethernet_disable_tx_rx_mac()....18
3.10. Ethernet_tx_rx_status_reg_read()....19
3.11. Ethernet_mode_reg_read()....19
3.12. Ethernet_tx_rx_control_reg_set()....20
3.13. Ethernet_get_statistic_counter()....20
3.14. Ethernet_print_statistic_counter()....21
3.15. Ethernet_print_all_statistic_counters()....21
4. API Data Structures....22
4.1. tsemac_reg_type_t....22
4.2. tsemac_handle_t....23
5. API Enum....24
5.1. Speed_mode_set....24
6. API Variables....25
7. API Macros....26
7.1. Success and Failure....26
7.2. IPG Time....26
7.3. Maximum Packet size....26
7.4. Control Register Macros....26
7.5. Mode Register Macros....26
7.6. Statistics Counters Registers Offset....26

7.7. Ethernet_reg_name_str(Offset) 27
References 28
Technical Support Assistance 29
Revision History 30

Figures

Figure 3.1. ethernet_init() 14
Figure 3.2. ethernet_packet_handle() 14
Figure 3.3. ethernet_set_mac_address() 15
Figure 3.4. ethernet_get_mac_address() 15
Figure 3.5. ethernet_set_multicast_address() 16
Figure 3.6. ethernet_get_multicast_address() 16
Figure 3.7. ethernet_set_speed() 17
Figure 3.8. ethernet_enable_tx_rx_mac() 18
Figure 3.9. ethernet_disable_tx_rx_mac() 18
Figure 3.10. ethernet_tx_rx_status_reg_read() 19
Figure 3.11. ethernet_mode_reg_read() 19
Figure 3.12. ethernet_tx_rx_control_reg_set() 20
Figure 3.13. ethernet_get_statistic_counter() 20
Figure 3.14. ethernet_print_statistic_counter() 21
Figure 3.15. ethernet_print_all_statistic_counters() 21

Tables

Table 1.1. Driver and IP Version 8
Table 1.2. Quick Facts on Driver Test Environment 8
Table 4.1. tsemac_handle_t Parameters 23
Table 5.1. speed_mode_set Enum Variables 24
Table 6.1. Variable Description 25

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
TSEMAC	Tri-Speed Ethernet Media Access Controller

1. Introduction

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

The TSEMAC 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 TSEMAC 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 TSEMAC extracts different components of a frame and transfers them to higher applications through the AXI4-stream interface.

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

1.1. Purpose

TSEMAC 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: 24.01.00.

1.4. Driver and IP Compatibility

Table 1.1. Driver and IP Version

Driver Version	IP Version
24.01.00	1.6.x

Table 1.2. Quick Facts on Driver Test Environment

Hardware Device	Tool Version
CertusPro-NX Versa Board	Lattice Propel™ Builder 2024.1
	Lattice Propel SDK 2024.1
	Lattice Radiant™ software 2024.1
	Radiant Programmer 2024.1

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.

```
uint8_t 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. • 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_packet_handle()

This API handles the transmitting and receiving of Ethernet packet by using shared memory and data mover. The same packet is looped back by the TSEMAC IP into shared memory.

```
void ethernet_packet_handle(tsemac_handle_t *handle,unsigned int *src_packet,unsigned int *dest_packet)
```

In/Out	Parameter	Description	Returns
In	handle	• The tsemac_handle_t structure contains the variables that need to be configured before passing it to ethernet_packet_handle. • frame_length - Indicates the length of the frame to be transferred from the source to the destination.	None
In/Out	src_packet	• Parameter for both input and output. • Represents the Ethernet packet to transmit to the TSEMAC.	
In/Out	dest_packet	• Parameter for both input and output. • Represents the Ethernet packet received from the TSEMAC.	

2.3. 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

2.4. 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.

```
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. • 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

2.5. 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

2.6. 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.

```
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. • 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

2.7. Ethernet_set_speed()

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

```
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.	0: failure 1: success

2.8. Ethernet_enable_tx_rx_mac()

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

```
unsigned int 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

2.9. Ethernet_disable_tx_rx_mac()

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

```
unsigned int 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

2.10. Ethernet_tx_rx_status_reg_read()

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

```
unsigned int ethernet_tx_rx_status_reg_read(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_tx_rx_status_reg_read().	Return the value of the transmit and receive status register.

2.11. Ethernet_mode_reg_read()

This API reads the TSEMAC mode register.

```
unsigned int ethernet_mode_reg_read(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_mode_reg_read().	Return the value of the mode register.

2.12. Ethernet_tx_rx_control_reg_set()

This API sets the bit for the control register.

```
unsigned char ethernet_tx_rx_control_reg_set(tsemac_handle_t *handle, unsigned char bit_pos)
```

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	bit_pos	Represents the bit position that the control register sets.	

2.13. 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.	

2.14. 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.	

2.15. 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 theTSEMAC.	None

3. Function Call Flow Diagrams

3.1. Ethernet_init()

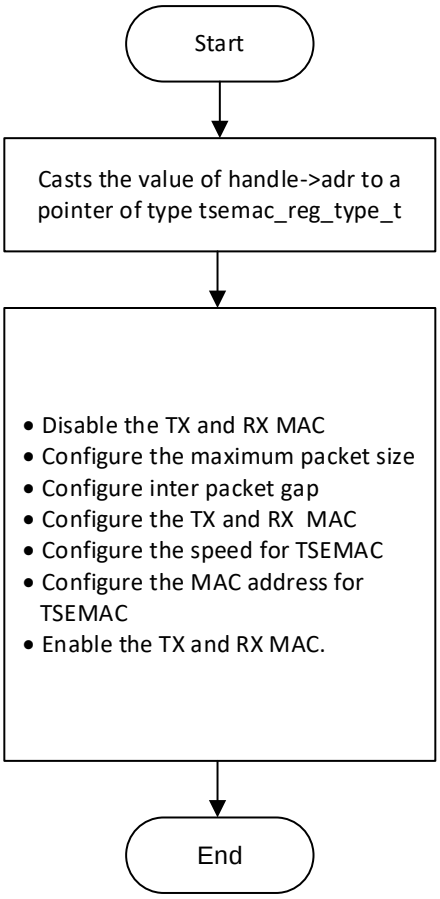

Figure 3.1. ethernet_init()

3.2. Ethernet_packet_handle()

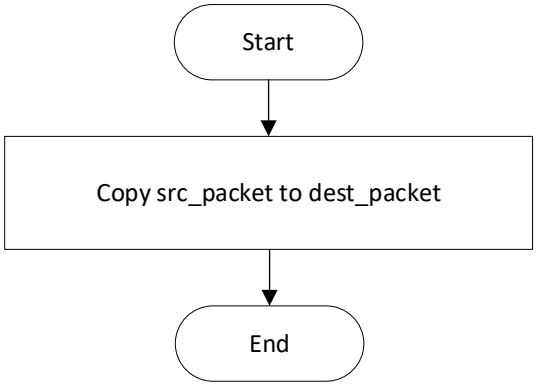

Figure 3.2. ethernet_packet_handle()

3.3. Ethernet_set_mac_address()

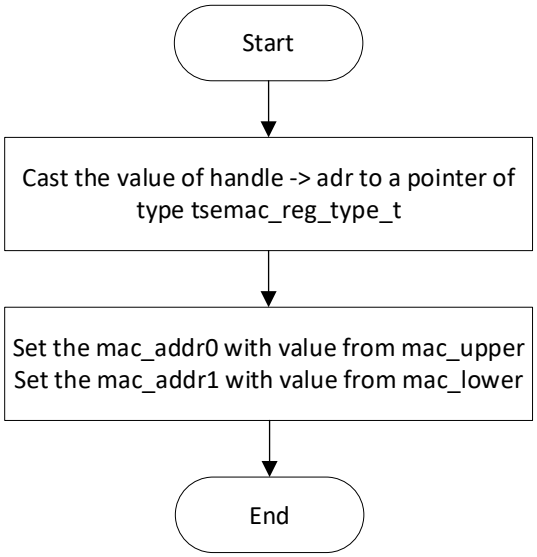

Figure 3.3. ethernet_set_mac_address()

3.4. Ethernet_get_mac_address()

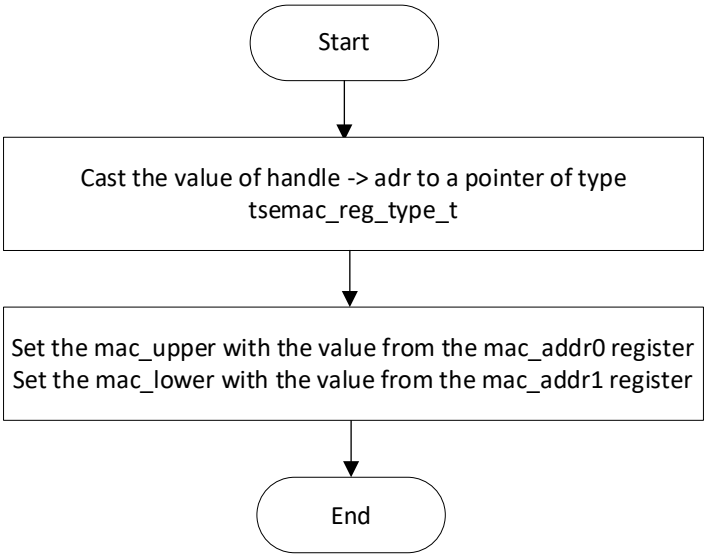

Figure 3.4. ethernet_get_mac_address()

3.5. Ethernet_set_multicast_address()

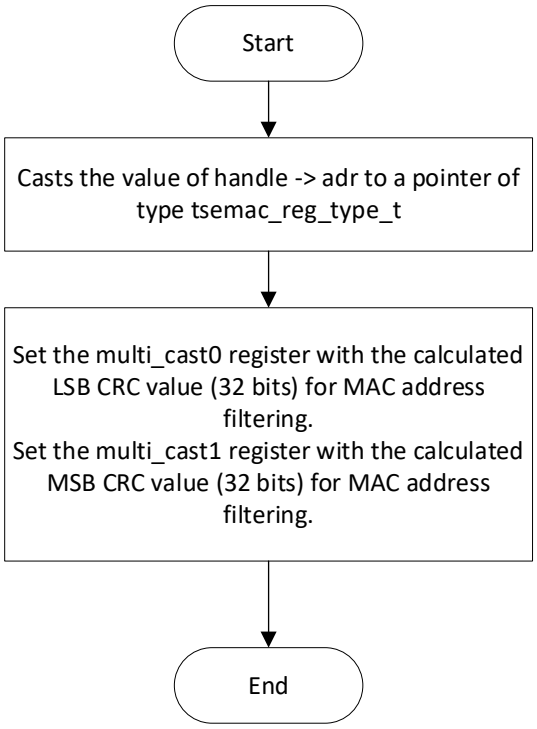

Figure 3.5. ethernet_set_multicast_address()

3.6. Ethernet_get_multicast_address()

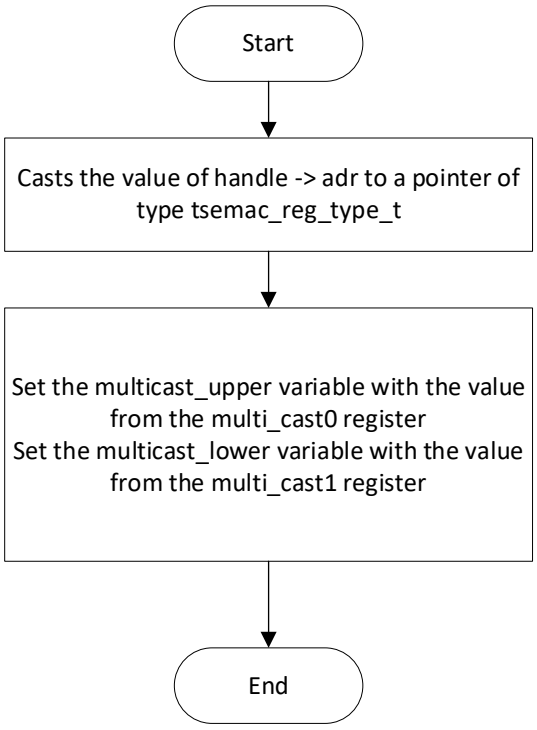

Figure 3.6. ethernet_get_multicast_address()

3.7. Ethernet_set_speed()

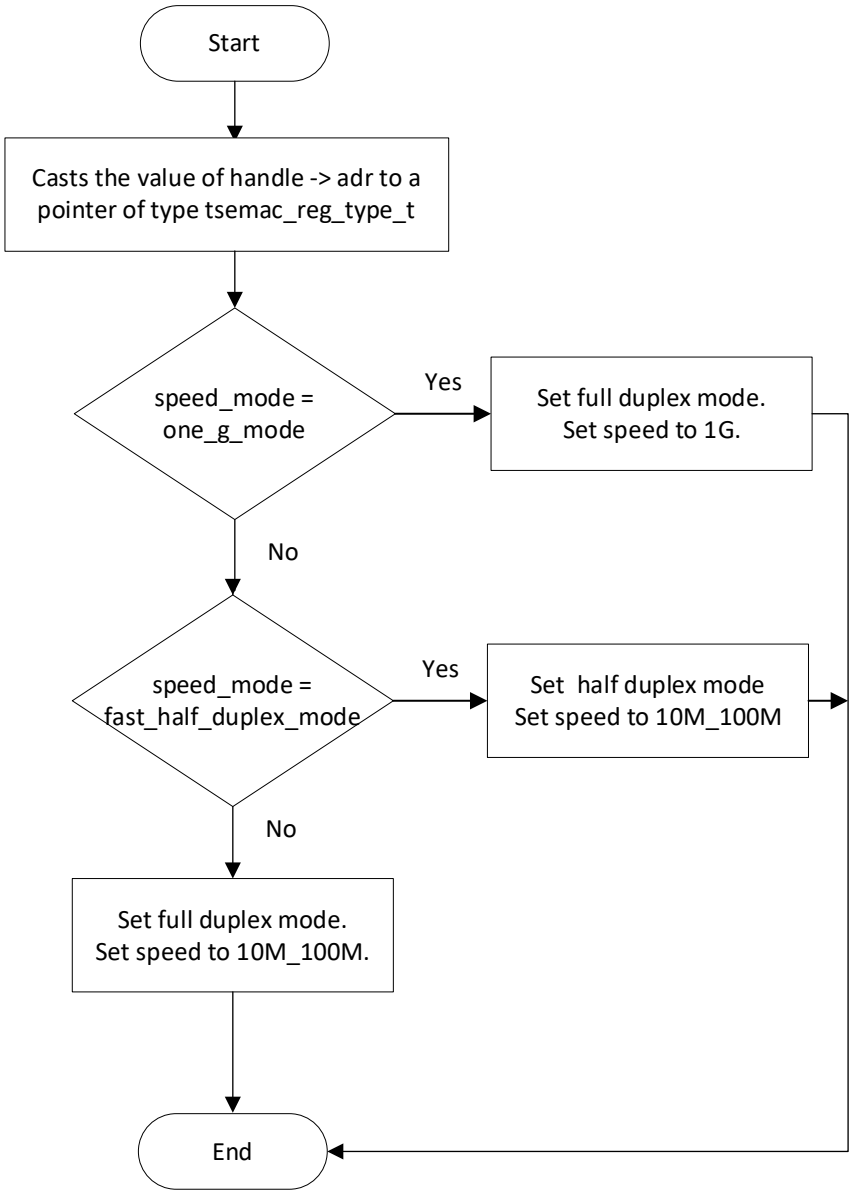

Figure 3.7. ethernet_set_speed()

3.8. Ethernet_enable_tx_rx_mac()

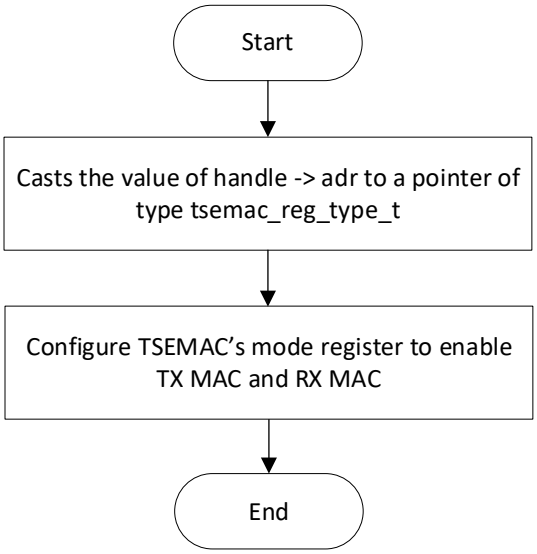

Figure 3.8. ethernet_enable_tx_rx_mac()

3.9. Ethernet_disable_tx_rx_mac()

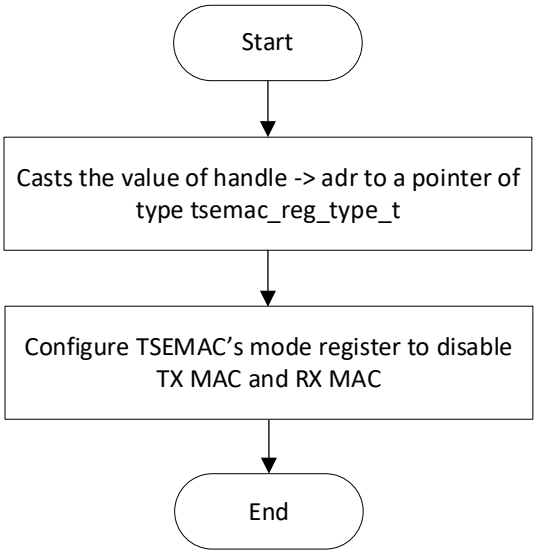

Figure 3.9. ethernet_disable_tx_rx_mac()

3.10. Ethernet_tx_rx_status_reg_read()

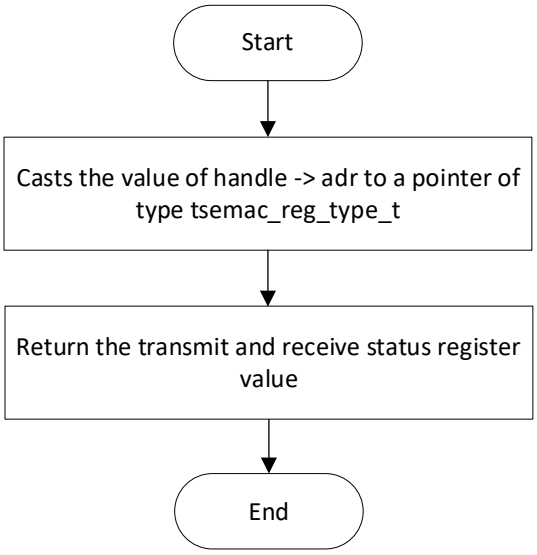

Figure 3.10. ethernet_tx_rx_status_reg_read()

3.11. Ethernet_mode_reg_read()

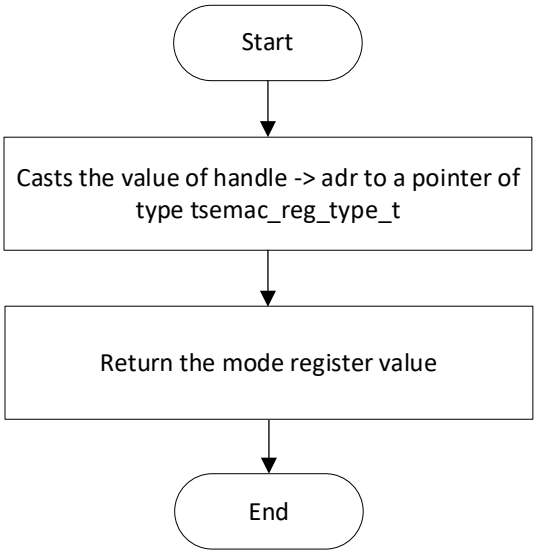

Figure 3.11. ethernet_mode_reg_read()

3.12. Ethernet_tx_rx_control_reg_set()

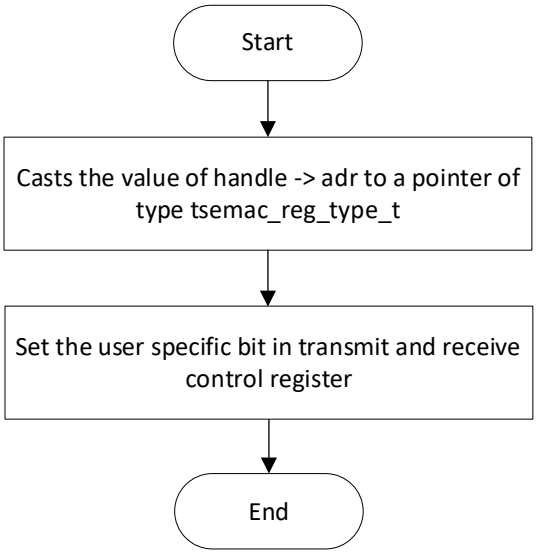

Figure 3.12. ethernet_tx_rx_control_reg_set()

3.13. Ethernet_get_statistic_counter()

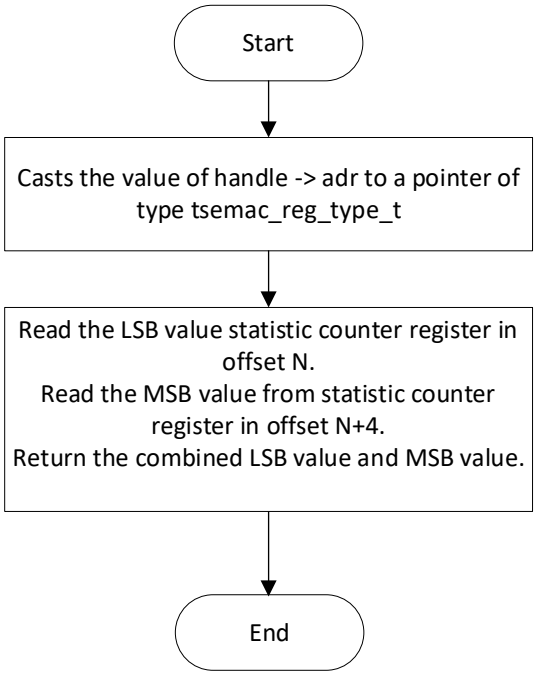

Figure 3.13. ethernet_get_statistic_counter()

3.14. Ethernet_print_statistic_counter()

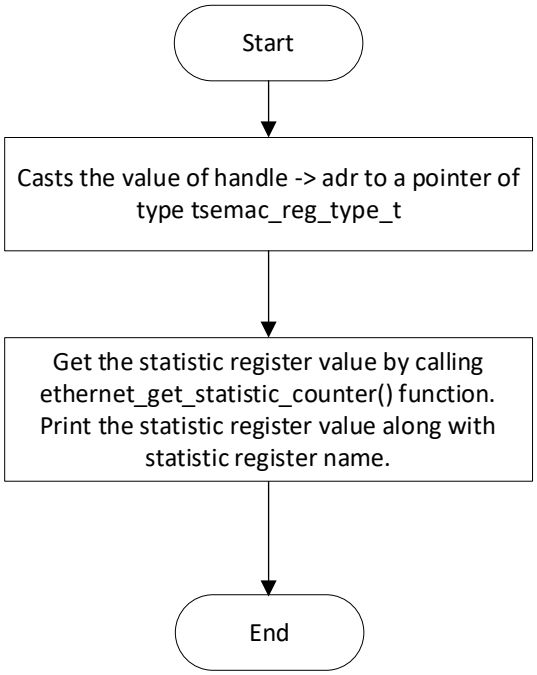

Figure 3.14. ethernet_print_statistic_counter()

3.15. Ethernet_print_all_statistic_counters()

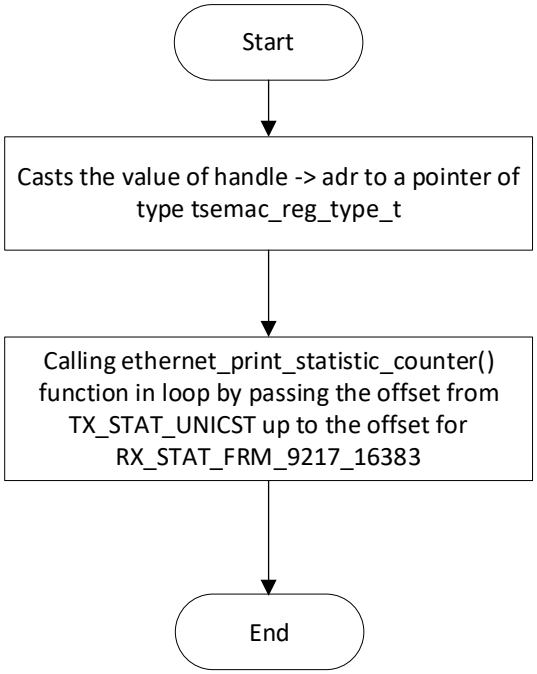

Figure 3.15. ethernet_print_all_statistic_counters()

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.
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.
enable_tx_mac	• Part of the tsemac_handle_t structure. • Used to enable the TX MAC.
enable_rx_mac	• Part of the tsemac_handle_t structure. • Used to enable the RX MAC.

5. API Enum

5.1. Speed_mode_set

This enum sets the different speeds and modes for TSEMAC. The following table shows the enum variables and description.

Table 5.1. speed_mode_set Enum Variables

Variable	Description
fast_half_duplex_mode	This variable sets the TSEMAC to half duplex mode and speed at 10/100 Mbps.
fast_full_duplex_mode	This variable sets the TSEMAC to full duplex mode and speed at 10/100 Mbps.
one_g_mode	This variable sets the TSEMAC to full duplex mode and speed at 1000 Mbps.

6. API Variables

The following table shows the variables and their description.

Table 6.1. 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.

7. API Macros

7.1. Success and Failure

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

7.2. IPG Time

```
#define       IPG_TIME                  12
```

7.3. 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.
```

7.4. Control Register Macros

```
#define       SET_FULL_DUPLEX_MODE      5
#define       SET_HALF_DUPLEX_MODE      5
```

7.5. Mode Register Macros

```
#define       SPEED_10_OR_100_MBPS      0
#define       SPEED_1G                  0
```

7.6. 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
```

7.7. 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 MAC IP Core User Guide (FPGA-IPUG-02084)
- 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 1.1, 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 Diagrams	• Updated the following figures:
	  • 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:

Section	Change Summary
	• Statistics Counters Registers Offset. • Ethernet_reg_name_str(Offset).

Revision 1.0, December 2023

Section	Change Summary
All	Initial release.

www.latticesemi.com