AN 029: Running 1000 Mbps TSE MAC with Pattern Generator

AN-029-v1.2
November 2021
www.efinixinc.com

Contents

Introduction

Efinix provides a Triple Speed Ethernet MAC (TSEMAC) core example design that targets the Trion® T120 BGA324 Development Board. The example design demonstrates the functionality of the TSEMAC core running at 1000 Mbps. The example design includes two test modes:

- *Normal mode test*—The MAC/UDP pattern generator constructs and initiates TX packets for the TX path of the TSEMAC core.
- *Link Partner mode test*—The TX packets are constructed by the other end of the network, for example a computer.

The example design consists of:

- *MAC/UDP pattern generator*—Generates the MAC/UDP TX packet to the TSEMAC core thru AXI4-ST TX interface.
- *RISC-V Jade SOC*—Allows you to configure the TSEMAC core configuration register and example design configuration register. Refer to the Table 6: Example Design Configuration Registers on page 15 for more details.
- *Triple Speed Ethernet MAC core*—Generated from Efinity® v2020.2 IP Manager (full package soft IP).

Figure 1: Example Design Block Diagram

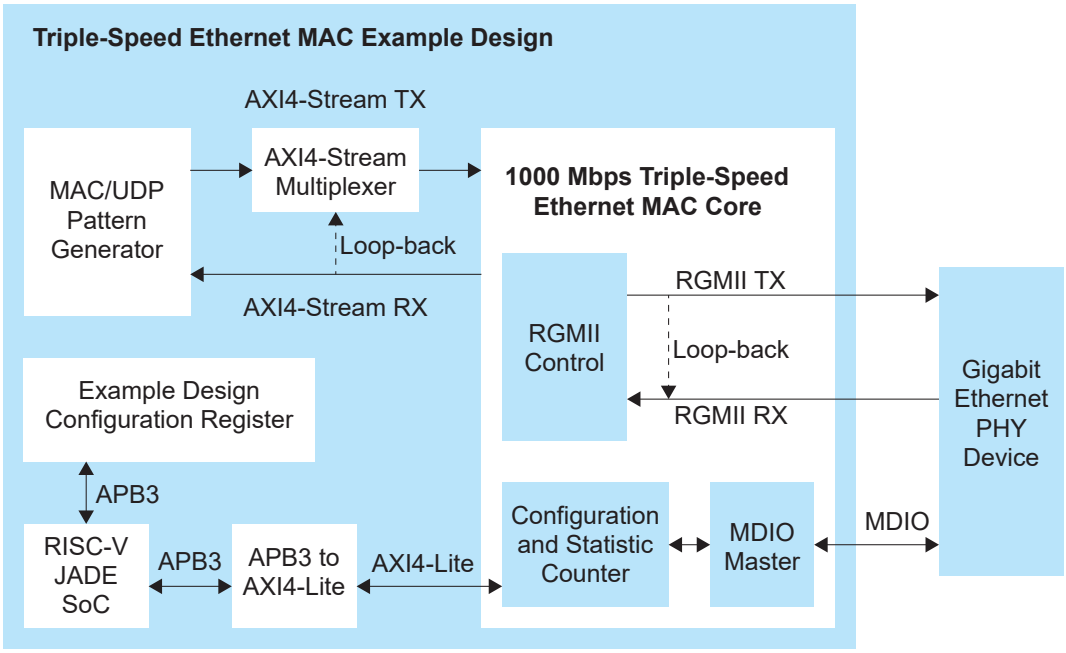


Table 1: Example Design Implementation

FPGA	LUTs	Memory Blocks	f_{MAX} (MHz)			Efinity® Version
			clk	clk_125m	rgmii_rxc	
T120 BGA324 C4	7,536	109	56.598	133.491	138.939	2020.2

Required Hardware

The example design uses the following hardware from the Trion® T120 BGA324 Development Kit:

- Trion® T120 BGA324 Development Board
- Micro-USB cable
- 12 V power adapter

You also need the following hardware, which you provide yourself:

- USB-to-UART converter module
- Ethernet cable
- Computer

Required Software

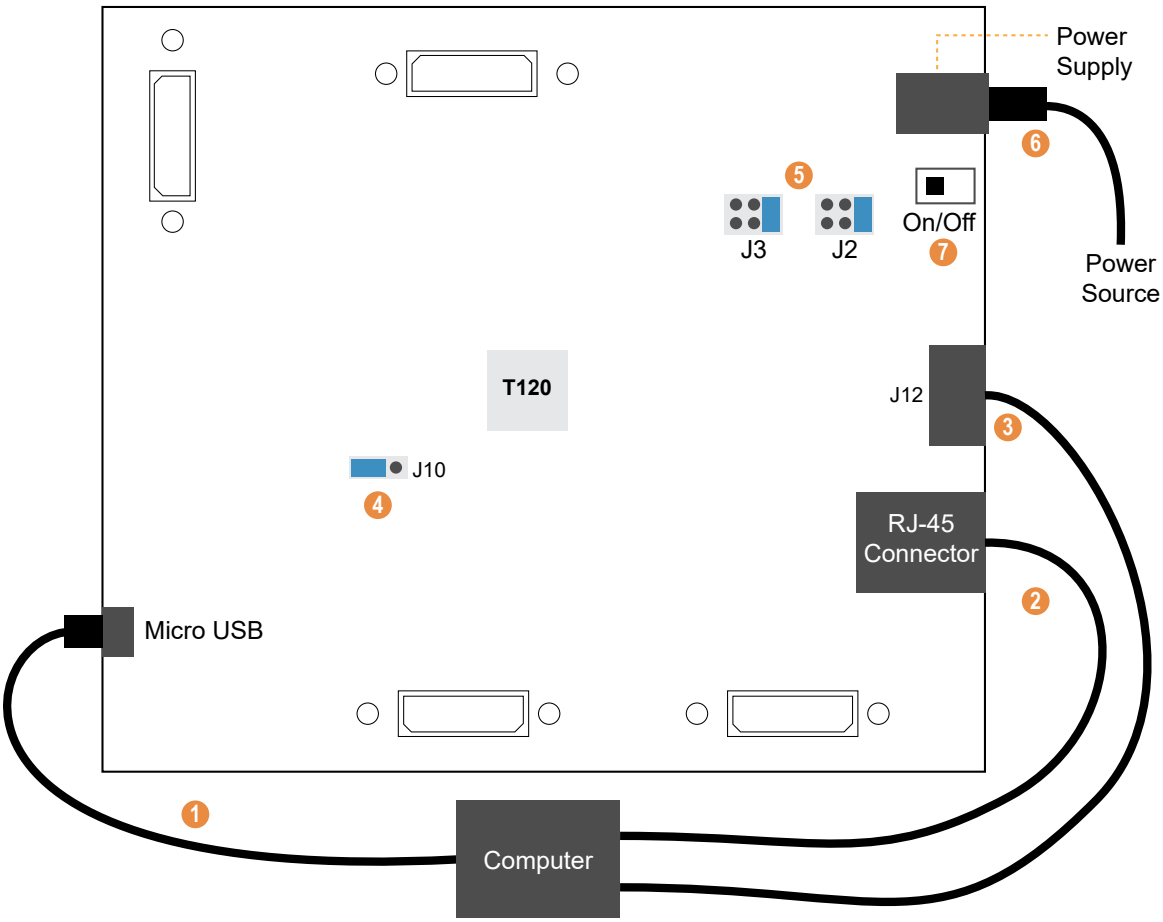
The example design uses the following hardware and software:

- RISC-V SDK for Windows or Ubuntu
- Efinity® software v2020.2

Set Up the Hardware

The following figure shows the hardware setup steps. If you have not already done so, attach standoffs to the board.

Figure 2: Hardware Setup



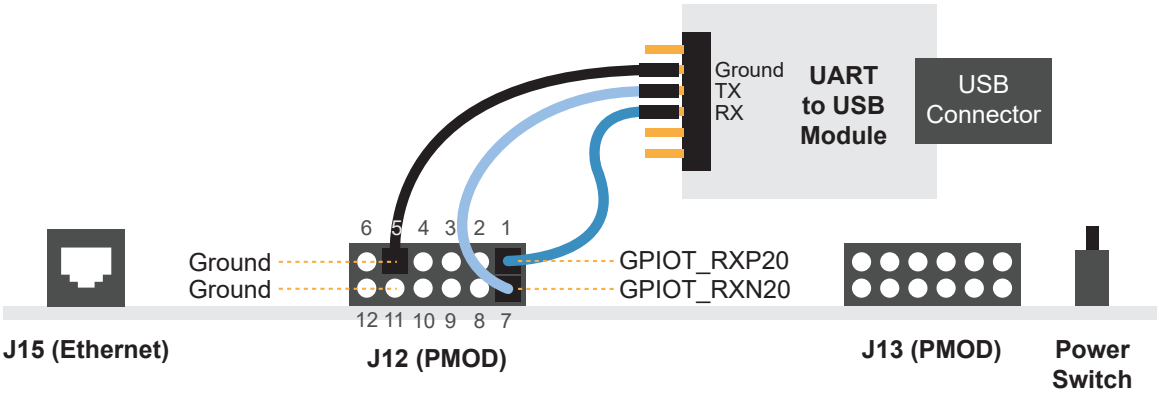
Important: Make sure that the Trion® T120 BGA324 Development Board is turned off before connecting any cables.

1. Connect a USB cable to the board and to your computer.
2. Connect an Ethernet cable to the RJ-45 jack on the board and to your computer.
3. Connect a USB-to-UART module to the J12 header. See Set Up a USB-to-UART Module (Trion) for more details.
4. On header J10, connect pins 2 - 3 with a jumper (default) to use the on-board oscillator.
5. Leave the jumper on J2 at the defaults (connecting pins 1 - 2). On J3, connect pins 1 - 2.
6. Connect the 12 V power supply to the board connector and to a power source.
7. Turn on the board using the power switch.

Set Up a USB-to-UART Module

A number of the software examples display messages on a UART terminal. The Trion® T120 BGA324 Development Board does not have a USB-to-UART converter, therefore, you need to use a separate USB-to-UART converter module. A number of modules are available from various vendors; any USB-to-UART module should work.

Figure 3: Connect the UART Module to I/O Header J12



1. Connect the UART's RX, TX, and ground pins to the Trion® T120 BGA324 Development Board using:
   - *RX*—GPIOT_RXP20, which is labeled as 1 on header J12
   - *TX*—GPIOT_RXN20, which is labeled as 7 on header J12
   - *Ground*—Ground, connect to one of the GND pins on header J12
2. Plug the UART module into a USB port on your computer. The driver should install automatically if needed.

Finding the COM Port (Windows)

1. Type Device Manager in the Windows search box.
2. Expand **Ports (COM & LPT)** to find out which COM port Windows assigned to the UART module; it is listed as USB Serial Port (COM*n*) where *n* is the assigned port number. Note the COM number.

Finding the COM Port (Linux)

In a terminal, type the command:

```
dmesg | grep ttyUSB
```

The terminal displays a series of messages about the attached devices.

```
usb <number>: <adapter> now attached to ttyUSB<number>
```

There are many USB-to-UART converter modules on the market. Some use an FTDI chip which displays a message similar to:

```
usb 3-3: FTDI USB Serial Device converter now attached to ttyUSB0
```

However, the Trion® T120 BGA324 Development Board also has an FTDI chip and gives the same message. So if you have both the UART module and the board attached at the same time, you may receive three messages similar to:

```
usb 3-3: FTDI USB Serial Device converter now attached to ttyUSB0
usb 3-2: FTDI USB Serial Device converter now attached to ttyUSB1
usb 3-2: FTDI USB Serial Device converter now attached to ttyUSB2
```

In this case the second 2 lines (marked by `usb 3-2`) are the development board and the first line (`usb 3-3`) is the UART module.

Program the Trion® T120 BGA324 Development Board

The core includes a bitstream file to get you started quickly. Download it to the board using these steps:

1. Download the file **tsemac_reference_design_IPM-v<version>.zip** from the Support Center.
2. Open the project (**temac_ex.xml**) in the Efinity software. The project is located in the **tsemac_reference_design_IPM-v<version>/tsemac/fpga/T120F324_devkit** directory.
3. Review the design.
4. Connect the Trion® T120 BGA324 Development Board to your computer using a USB cable.
5. Use the Efinity® Programmer to download the bitstream file to your board. The bitstream file is in the **tsemac_reference_design_IPM-v<version>/tsemac/fpga/T120F324_devkit** directory.

Learn more: Instructions on how to use the Efinity® software and board documentation are available in the Support Center.

Using this Example with the RISC-V SDK

Before working with the software included with this example design, you should already be familiar with using the Jade SoC and RISC-V SDK. Specifically, you should know how to:

- Launch Eclipse using the **run_eclipse.bat** file (Windows) or **run_eclipse.sh** file (Linux) scripts.
- Set up global environment variables.
- Create and build a software project.
- Debug using the OpenOCD debugger.
- Open a UART terminal.

Learn more: Refer to the Jade RISC-V SoC Hardware and Software User Guide for detailed instructions on how to perform these tasks.

Setting the Double Data I/O

The Triple Speed Ethernet MAC can run at 10, 100, and 1000 Mbps Ethernet speed. You must have the double data I/O (DDIO) option enabled on certain signals for the Triple Speed Ethernet MAC core to run at 1000 Mbps.

Because the Trion® T120 BGA324 Development Board has a limited number of available GPIO with capability, the example design does not assign the TSEMAC RGMII control signals, `rgmii_tx_ctl` and `rgmii_rx_ctl` to a DDIO block. Instead, the HI and LO TX control signals are OR together, while the HI and LO RX control signal are assigned with the same input control signal. You do not need this logic if you are not using a Trion® T120 BGA324 Development Board or a T120 BGA576 Development Board, and the hard DDIO block is available for `rgmii_tx_ctl_HI`/`rgmii_tx_ctl_LO` and `rgmii_rx_ctl_HI`/`rgmii_rx_ctl_LO`.

To disable the logic, comment out the following lines in the **tsemac_ex.v** file.

```
assign rgmii_tx_ctl = rgmii_tx_ctl_HI | rgmii_tx_ctl_LO;
assign rgmii_rx_ctl_HI = rgmii_rx_ctl;
assign rgmii_rx_ctl_LO = rgmii_rx_ctl;
```

Note: The project file included in the example design has predefined all necessary Interface Designer settings. You do not have to set the Interface Designer described in this section again. You can refer to the following settings when designing your own Triple Speed Ethernet MAC core.

Interface Designer Settings

You must enable the DDIO option for the following signals:

- `rgmii_txc`

In the Efinity® Interface Designer, use the Create Block menu to add the signals and set the Block Editor settings as shown in the following table.

Table 2: Block Editor Settings for rgmii_txc

Parameter	rgmii_txc
Instance Name	User defined
Mode	output
Register Option	register
Double Data I/O Option	resync
Clock Pin Name	-
Pin Name (HI)	rgmii_txc_HI
Pin Name (LO)	rgmii_txc_LO
Enable invert clock	Yes
Output Clock Pin Name	clk_125m_90deg

You must enable the DDIO option for the following buses:

- `rgmii_rxd`
- `rgmii_txd`

Use the Create GPIO Bus wizard to add the buses and set the Bus Property settings as shown in the following table.

Table 3: Bus Property Settings for rgmii_rxd and rgmii_txd

Parameter	Setting	
	rgmii_rxd	rgmii_txd
Mode	Input	Output
Pin Name (HI)	rgmii_rxd_HI	rgmii_txd_HI
Pin Name (LO)	rgmii_rxd_LO	rgmii_txd_LO
Register Option	register	register
Double Data I/O Option	resync	resync
Clock Pin Name	rgmii_rxc	-
Output Clock Pin Name	-	clk_125m
Enable invert clock	No	No

Double Data I/O Waveforms

The following figures show the input and output waveforms when the DDIO option are enabled in the Triple Speed Ethernet MAC core signals.

Figure 4: Output Signals Waveform with DDIO Enabled

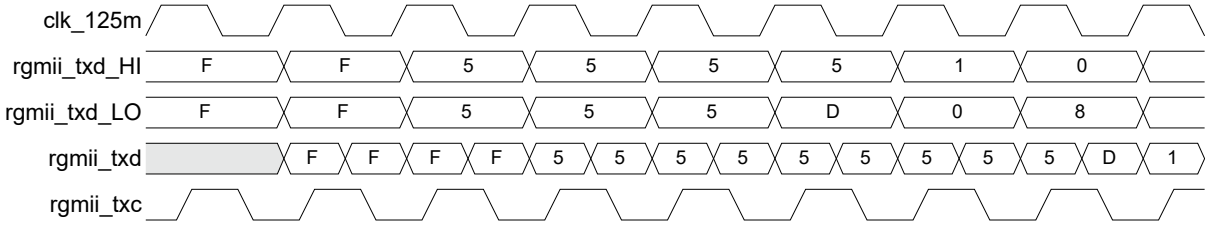
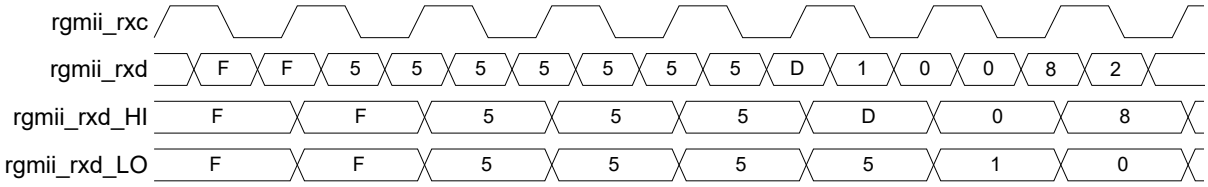


Figure 5: Input Signals Waveform with DDIO Enabled



The Trion® T120 BGA324 and Trion® T120 BGA576 development boards have a Davicom PHY IC. You may need to use different DDIO settings when using different PHY IC with a different transit behaviour.

The Davicom PHY IC transits the `rgmii_rxd` and `rgmii_rx_ctl` at the rising edge of `rgmii_rxc`. The Triple Speed Ethernet MAC receives the first data, `rgmii_rxd`, at the falling edge of `rgmii_rxc`. With this receiving pattern, the DDIO for `rgmii_rxd` and `rgmii_rx_ctl` is set to resync mode.

If you use a PHY IC that transits the data, `rgmii_rxd`, at the rising edge of `rgmii_rxc`, you must make one of the following design adjustments:

- Set the DDIO to normal mode. Swap the `rgmii_rxd_HI` to `rgmii_rxd_LO`, and the `rgmii_rxd_LO` to `rgmii_rxd_HI` when instantiating the core.
- Set the PHY RX clock-to-data delay of the RGMII interface in the software driver code to align to the falling edge of RX clock to the data.

Note: Refer to Setting PHY RGMII Clock-to-Data Delay in the **AN 029: Running 1000 Mbps TSE MAC with Pattern Generator** on how to change the PHY RGMII clock-to-data delay in the software driver code (**phy.h**).

Example Design Test Modes

The example design includes software for the Jade SoC in the **tsemac_software** directory. In addition to the source code, Efinix provides compiled binary files using predefined settings so you can easily try out two test modes:

- *Normal Test Mode*—Use the **mac_debug.elf** file in the **build\test_mode_0** directory.
- *Link-partner Test Mode*—Use the **mac_debug.elf** file in the **build\test_mode_1** directory.

You can also customize the test modes by changing the variables in the source code's **main.h** file. However, you must rebuild the software if you change any variable.

Figure 6: Defining Variables in main.h File

```
/************************** Project Header File ***************************/
#define PRINTF_EN   1
#define TEST_MODE   1//0:Normal Mode; 1:Link partner Test Mode;

#define PAT_NUM      0
#define PAT_DLEN     8
#define PAT_IPG      4095//4095//255
#define PAT_TYPE     0//0:UDP Pattern; //1:MAC Pattern;
#define DST_MAC_H    0xffff
#define DST_MAC_L    0xffffffff
#define SRC_MAC_H    0xeae8
#define SRC_MAC_L    0x5e0060c8
#define SRC_IP       0xc0a80164
#define DST_IP       0xc0a80165
#define SRC_PORT     0x0521
#define DST_PORT     0x2715
```

Setting PHY RGMII Clock-to-Data Delay

In cases where you want to re-align the PHY clock with respect to the data to allow the TSEMAC core to receive or transmit data correctly, you can set the PHY RGMII clock-to-data delay, `TX_delay` and `RX_delay`, in the software driver function code, **phy.h** as below:

Figure 7: Defining Delay in Software Function Code

```
/************************** Function File ***************************/
void PhyDlySetRXTX(int RX_delay, int TX_delay)
{
    u32 Value;
    StrPrintf("Start Info : Set Phy Delay.\r\n");
    Phy_Wr(0x1F,0x0168);
    Phy_Wr(0x1E,0x8040);
    Phy_Wr(0x1E,0x401E);
    Value = Phy_Rd(0x1F) & 0xFFFF;

    Value &= 0xFF00;
    RX_delay &= 0xF;
    TX_delay &= 0xF;
    P1Printf("Setup New Value = \r\n", RX_delay);

    Value = ((Value) | (RX_delay<<4) | (TX_delay));

    Phy_Wr(0x1F,Value);
    Phy_Wr(0x1E,0x801E);

    Phy_Wr(0x1E,0x401E);
    Value = Phy_Rd(0x1F) & 0xFFFF;
    P1Printf("Read New Value = \r\n", Value);
}
```

Table 4: RX_delay and TX_delay Delay Settings

Delays	Values	Description
RX_delay and TX_delay	0 – 15	0: No delay 15: 4 ns delay Each step is equivalent to 0.267 ns delay.

Normal Mode Test

The MAC/UDP pattern generator constructs and initiates TX packets for the TX path of the TSEMAC core. The receiving end can be either at other TSEMAC RX (disable RGMII interface loop back) or looped-back at the internal RGMII control block (enable RGMII interface loop back).If the RGMII interface loop back is disabled, the TSEMAC RX path receives the RX packet from the other TSEMAC TX at the other end or from Ethernet PHY device loop-back. The simulation testbench included in this example uses the loop-back method.

Table 5: Normal Mode Macros

Macro Name	Description
PAT_NUM	Number of transmit packet. Unlimited packet transmission is denoted by 0.
PAT_DLEN	Length of the UDP/MAC pattern transmit packet.
PAT_IPG	Number of interpacket gap. One interpacket gap is 8 ns.
PAT_TYPE	Type of transmit packet pattern. 1: MAC pattern 0: UDP pattern

The test passes when:

- The value of `PAT_NUM` is 0, the value of `aFramesTransmittedOK` continues to increase until 0xffffffff. If the value of `PAT_NUM` is non-zero, the value of `aFramesTransmittedOK` and `PAT_NUM` must be the same.
- The value of `aFramesReceivedOK`is the same with the number of frames sent by the opposite end of the network. The `ifInErrors` is 0.
- The captured network packet is consistent with the transmitted packet pattern. See Using Wireshark on page 15 for more information.

Note: You can use any terminal program, such as Putty, termite, or the built-in Eclipse terminal to monitor the `aFramesTransmittedOK`, `aFramesReceivedOK`, and `ifInErrors` statistic counter register values.

Figure 8: Normal Mode Telnet Session Output Example

```
---EFX-RISCV Command Line Menu Demo---
Wait Ethernet Link up...
Rd Phy Addr 0 : 1140
Rd Phy Addr 2 : 6e
Rd Phy Addr 3 : 3212
Wr Phy Addr 1f : 168
Wr Phy Addr 1e : 8040
Start Info : Set Phy Delay.
Wr Phy Addr 1e : 401e
Rd Phy Addr 1f : 5988
Setup New Value =
 : 0
Wr Phy Addr 1f : 5900
Wr Phy Addr 1e : 801e
Wr Phy Addr 1e : 401e
Rd Phy Addr 1f : 5900
Read New Value =
 : 5900
Rd Phy Addr 11 : ac00
Info : Phy Link up on 1000Mbps.
Info : Set Mac Speed.
Info : Set Mac IPG.
Info : Assert mac reset
Info : Deassert mac reset
Info : Mac Reset Statistics Counters.
Info : Set Pattern Generator.
--------------------
aFramesTransmittedOK : 15c3e
aFramesReceivedOK : 3
ifInErrors : 0
ifOutErrors : 0
etherStatsPkts : 3
etherStatsUndersizePkts : 0
etherStatsOversizePkts : 0
aRxFilterFramesErrors : 0
aFrameCheckSequenceErrors : 0
aTxPAUSEMACCtrlFrames : 0
aRxPAUSEMACCtrlFrames : 0
--------------------
--------------------
aFramesTransmittedOK : 2bb29
aFramesReceivedOK : 5
ifInErrors : 0
ifOutErrors : 0
etherStatsPkts : 5
etherStatsUndersizePkts : 0
etherStatsOversizePkts : 0
aRxFilterFramesErrors : 0
aFrameCheckSequenceErrors : 0
aTxPAUSEMACCtrlFrames : 0
aRxPAUSEMACCtrlFrames : 0
--------------------
```

Link-Partner Test

The TX packets are constructs by the other end of the network, for example, a computer. The AXI4-ST loop back is enabled in this test mode, and the MAC/UDP pattern generator is not used. The TSEMAC core receives the TX packet from the PC and transmits the received packet back to the PC at the AXI4-ST interface.

The test passes when:

- The value of `aFramesTransmittedOK` and `aFramesReceivedOK` is consistent.
- The `ifInErrors`is 0.
- The captured network packet is consistent with the transmitted packet pattern. See **Using Wireshark** on page 15 for more information.

Note: You can use any terminal program, such as Putty, termite, or the built-in Eclipse terminal to monitor the `aFramesTransmittedOK`, `aFramesReceivedOK`, and `ifInErrors` statistic counter register values.

Figure 9: Link-Partner Mode Telnet Session Output Example

```
---EFX-RISCV Command Line Menu Demo---
Wait Ethernet Link up...
Rd Phy Addr 0 : 1140
Rd Phy Addr 2 : 6e
Rd Phy Addr 3 : 3212
Wr Phy Addr 1f : 168
Wr Phy Addr 1e : 8040
Start Info : Set Phy Delay.
Wr Phy Addr 1e : 401e
Rd Phy Addr 1f : 5988
Setup New Value =
 : 0
Wr Phy Addr 1f : 5900
Wr Phy Addr 1e : 801e
Wr Phy Addr 1e : 401e
Rd Phy Addr 1f : 5900
Read New Value =
 : 5900
Rd Phy Addr 11 : ac00
Info : Phy Link up on 1000Mbps.
Info : Set Mac Speed.
Info : Set Mac IPG.
Info : Assert mac reset
Info : Deassert mac reset
Info : Set Mac Address.
Info : Mac Reset Statistics Counters.
--------------------
aFramesTransmittedOK : 3
aFramesReceivedOK : 3
ifInErrors : 0
ifOutErrors : 1
etherStatsPkts : 3
etherStatsUndersizePkts : 0
etherStatsOversizePkts : 0
aRxFilterFramesErrors : 0
aFrameCheckSequenceErrors : 0
aTxPAUSEMACCtrlFrames : 0
aRxPAUSEMACCtrlFrames : 0
--------------------
--------------------
aFramesTransmittedOK : e
aFramesReceivedOK : e
ifInErrors : 0
ifOutErrors : 1
etherStatsPkts : e
etherStatsUndersizePkts : 0
etherStatsOversizePkts : 0
aRxFilterFramesErrors : 0
aFrameCheckSequenceErrors : 0
aTxPAUSEMACCtrlFrames : 0
aRxPAUSEMACCtrlFrames : 0
--------------------
--------------------
aFramesTransmittedOK : 14
aFramesReceivedOK : 14
ifInErrors : 0
ifOutErrors : 1
etherStatsPkts : 14
etherStatsUndersizePkts : 0
etherStatsOversizePkts : 0
aRxFilterFramesErrors : 0
aFrameCheckSequenceErrors : 0
aTxPAUSEMACCtrlFrames : 0
aRxPAUSEMACCtrlFrames : 0
--------------------
```

Using Wireshark

Wireshark is a free, open-source network packet analyzer software. Efinix® recommends that you use Wireshark to check for captured and transmitted packet consistency when the opposite end of the network is connected to a PC. You can download Wireshark from www.wireshark.org.

Example Design Configuration Registers

Table 6: Example Design Configuration Registers

Default value for all DWORD offsets are 0x0.

DWORD Offset	Name	R/W	Description
0x200	mac_sw_rst	R/W	Assert high to trigger proto_reset signal. The register output is tied to proto_reset signal.
0x204	Bit 0 - axi4_st_mux_select	R/W	0: Transmit AXI4-Stream TX packet from pattern generator. 1: Loopback AXI4-Stream RX path to AXI4-Stream TX path.
	Bit 1 - pat_mux_select	R/W	0: Select UDP pattern generator. 1: Select MAC pattern generator.
0x208	Bit 0 - udp_pat_gen_en	R/W	Assert high then low to trigger a UDP packet.
	Bit 1 - mac_pat_gen_en	R/W	Assert high then low to trigger MAC packet.
0x20c	Bit [15:0] - pat_gen_num	R/W	The number of transmit packet to send. Zero indicates unlimited packets.
	Bit [31:16] - pat_gen_ipg	R/W	The UDP/MAC interpacket gap.
0x210	pat_dst_mac [31:0]	R/W	The UDP/MAC pattern's destination MAC address.
0x214	pat_dst_mac [47:32]	R/W	
0x218	pat_src_mac [31:0]	R/W	The UDP/MAC pattern's source MAC address.
0x21c	pat_src_mac [47:32]	R/W	
0x220	pat_mac_dlen [15:0]	R/W	The length of a MAC pattern packet. The complete MAC pattern packet is pat_mac_dlen + 14 bytes (excluding CRC).
0x224	pat_src_ip [31:0]	R/W	The UDP pattern's source IP address.
0x228	pat_dst_ip [31:0]	R/W	The UDP pattern's destination IP address.
0x22c	Bit [15:0] pat_src_port	R/W	The UDP pattern's source port.
	Bit [31:16] pat_dst_port		The UDP pattern's destination port.
0x230	pat_udp_dlen [15:0]	R/W	The length of a UDP pattern packet. The complete UDP pattern packet is pat_udp_dlen + 45 bytes (excluding CRC).

Revision History

Table 7: Revision History

Date	Version	Description
November 2021	1.2	Removed Block Editor Settings for rgmii_rx_ctl and rgmii_tx_ctl signals.
October 2021	1.1	Updated Block Editor Settings for rgmii_rx_ctl and rgmii_tx_ctl signals. (DOC-586)
March 2021	1.0	Initial release.