Application Note

Basic Ethernet Interface Debug With Linux®

Schuyler Patton, Daolin Qiu, Alvaro Reyes

ABSTRACT

This application note provides guidance on resolving basic board-level Ethernet connection issues as well as basic transmission and reception of packets encountered on TI processors. The examples explained within are in reference to Texas Instruments' Processor SDK Linux® on custom boards or TI EVMs. The main takeaway of this application note is how to verify basic transmission and reception of error-free packets at the MAC level.

Table of Contents

1 Terminology....2
2 Prerequisites....2
3 Quick Initial Steps....3
4 Debug Overview....5
5 Network Driver Initialization Process....6
6 Ethernet PHY Analysis....8
7 Ethernet MAC Statistics Analysis....10
8 How IPv4 Address is Obtained....12
9 Follow the Packet....13
10 Debug Network Topologies and Techniques....15
10.1 Direct Connection....15
10.2 Static IP Addresses....15
10.3 Setting Low Bit Rates....15
10.4 Beware When Connecting to a Switch....16
11 Linux® Utilities Summary....17
12 Checklist for Requesting Ethernet Support....18

List of Figures

Figure 3-1. Full Ethernet Packet Flow....3
Figure 4-1. Follow the Packet Flow....5
Figure 4-2. Ethernet MAC and Ethernet PHY Interface....6
Figure 6-1. RGMII Receive Multiplexing and Timing Diagram....9
Figure 7-1. Full Ethernet Packet Flow With MAC Statistics....10
Figure 8-1. Wireshark DHCP Messages....12
Figure 9-1. DUT Transmit Path....13
Figure 9-2. DUT Receive Path....14
Figure 10-1. Direct Connection Topology....15
Figure 10-2. Wireshark Ping Messages....16

Trademarks

1 Terminology

CPSW	Common Platform Switch, a TI-designed Ethernet MAC typically used for general-purpose Ethernet applications
PRU_ICSSG	Programmable Real-Time Unit - Industrial Communications Subsystem Gigabit, a TI-designed Ethernet MAC typically used in industrial Ethernet applications with strict low-latency requirements
PHY	Ethernet PHY or Ethernet physical transceiver
MAC	Ethernet MAC or Ethernet media access controller
DTS	Device Tree Source
CRC	Cyclic Redundancy Check, a type of checksum algorithm used by Ethernet communication, CRC errors show up when a device receives data that is corrupted
Porting	In the context of this application note, porting means process of configuring the Linux Board DTS
DUT	Device Under Test
Link Partner	Describes the device directly connected to the DUT
Direct Connect	Describes a network topology with two directly connected platforms
Follow-the-packet	A debug technique introduced in this application note where the engineer uses the MAC hardware statistics to follow the path that an Ethernet packet goes through for transmission and reception at the wire level
MPU	Microprocessor Unit

2 Prerequisites

The following list includes details on the required setup prior to debugging an Ethernet interface on a TI EVM or a custom board:

- When two platforms with Ethernet interfaces are connected to each other, the platform or device under test is called the DUT and the other platform is considered to be a link partner
- The DUT can be a custom board or a TI EVM
- If the issue is on a custom board, try the same test on a TI EVM that is running the latest TI SDK
- For best results, use the latest TI Software Development Kit (SDK) for the DUT
- Use the Linux user console on either a TI EVM or the custom board
- The engineer performing the debugging knows how to use a Linux console
- Only work with one interface at a time to keep the debug as simple as possible
- The DUT is connected to a known valid link partner such as a PC or another board that has been proven to work
- The DTS file is configured such that the DUT has successfully booted to a Linux user command prompt
- If using an Ubuntu® PC as the link partner, do not run the PC as a virtual machine because it is possible that the resulting Ethernet MAC statistics are not accurate

3 Quick Initial Steps

Before debugging a potential network connection issue, there are some initial steps to verify if there is an issue with basic transmission or reception of Ethernet packets. The first step is to see if there is an Ethernet interface defined. The next step is to determine if a link was established between two link partners. The final step is verifying if Ethernet MAC statistics show packets are being transmitted or received without any corruption.

The following is a summary of the first steps in debugging the Ethernet interface. Perform these steps after the DUT has successfully booted into a Linux environment.

1. Check if the interface has initialized by using `ifconfig -a` or `ip link show`
2. Determine if a link was established with a link partner by using `ethtool <interface name>`
   - On TI EVMs this is `ethtool eth0` to view the details for the first Ethernet interface
3. Verify if any packets have been sent or received by using `ethtool -S <interface name>`

The term *link partners* simply means two directly connected platforms such as what is shown Figure 3-1. This is called a *direct connect* topology. For the above three initial steps, a recommended link partner to the DUT is a PC running Ubuntu or another known good Linux platform. The reason why the recommended link partner is on a Linux platform is to provide a symmetric test environment such that the link partner mirrors a similar platform to that of the DUT running Linux. Additionally, `ethtool` is easily available on a Linux platform.

Figure 3-1 shows how the Ethernet interface is broken down into two components: the MAC and the PHY. The MAC is responsible for sending and receiving packets. The PHY is responsible for converting the signals from the MAC to the Ethernet cable.

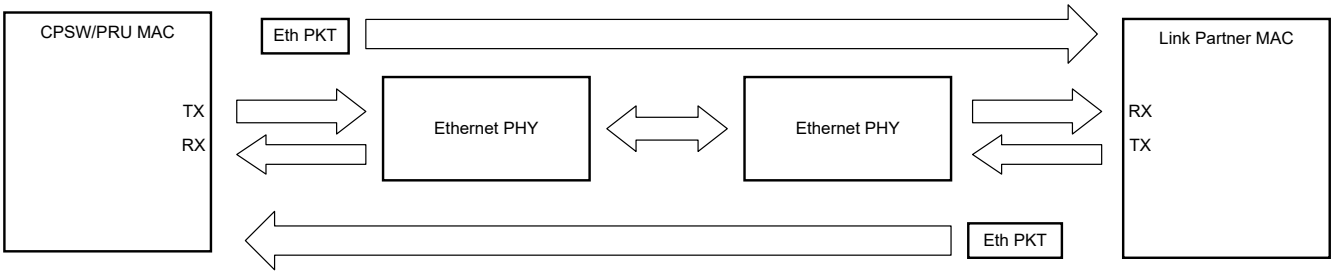


Figure 3-1. Full Ethernet Packet Flow

After confirming the interface to test is listed with `ifconfig -a` or `ip link show`, the next step is to see if there is an Ethernet link detected with a link partner.

Note

Debug Suggestions:

If the interface is not listed from the `ifconfig` or `ip a` command, then this can mean there is an error in the DTS file that defines the CPSW or PRU-ICSSG Ethernet interface. Both CPSW and PRU-ICSSG are TI-designed Ethernet interfaces. See also Section 5.

When an Ethernet cable is connected between the two link partners such as a TI EVM and a PC, the Ethernet PHY on both devices attempts to establish a link between the two devices independent of any software running such as a `ping` test. If there is a link detected, the next step is to check if any traffic has occurred for either reception of packets or transmission of packets. If the Ethernet driver on the DUT does not see a link detection, then the Ethernet driver does not initialize and no network traffic passes.

The `ethtool` utility is used for both steps of link detection and traffic analysis. Step 1 is checking for link detection. In this example, the interface under test is assumed to be *eth0*. The following code snippet shows a partial output of the `ethtool` command. There is a lot of information printed out but the section of interest is the `Link detected:` property. The `Linked detected:` is either *yes* if a link is detected or *no* if a link is not

detected. The speed and duplex mode are also returned by this command and indicate what the link speed and duplex mode are.

```
root@am62xx-evm:~# ethtool eth0
Settings for eth0:
        ...
        Speed: 1000Mb/s
        Duplex: Full
        ...
        PHYAD: 0
        Link detected: yes
```

If a link is not detected, then no traffic is able to pass. Resolve this situation before proceeding.

Note

Debug Suggestions:

If no link is detected by the connected PHY, then the next step is to review the PHY hardware configuration and MDIO communication information explained in Section 6.

After an Ethernet link is detected, the MAC hardware statistics (results from `ethtool -S <interface name>`) can be reviewed. The information returned from the MAC hardware statistics contains data about basic packet send and receive. Examining if packets are sent and received on the DUT gives an indication to whether the Ethernet interface and Ethernet PHY is functioning or not. Additionally, the DUT not receiving an IPv4 address or not receiving a ping response does not necessarily mean that there is an issue with the basic sending and receiving of packets. For example, not receiving an IPv4 address can be as simple as there not being a DHCP server to provide the IPv4 address.

The Ethernet MAC statistics are the most accurate set of statistics in terms of whether the Ethernet MAC is sending and receiving packets. These are the most accurate statistics to look at because this data is directly measured from the physical wire, with no additional abstraction layers. The transmitted (*TX packets*) or received (*RX packets*) packet counts shown by the `ifconfig` or `ip -s link` command is not completely accurate from a MAC perspective. Packets can be dropped between the kernel network stack and the MAC driver.

The following code snippet shows a partial example of Ethernet MAC hardware statistics showing received and transmitted packets as well as any associated errors. This example shows what is expected when there is good reception of Ethernet frames being detected and successfully transmitted Ethernet frames. There can be transmit errors that are only be viewable by the connected link partner Ethernet MAC hardware statistics. More details on link partner analysis are found in Section 9.

```
root@am62xx-evm:~# ethtool -S eth0
NIC statistics:
         ...
         rx_good_frames: 1127
         ...
         rx_crc_errors: 0
         rx_align_code_errors: 0
         ...
         tx_good_frames: 163
```

Find more information on how RX and TX traffic errors can be analyzed in Section 7.

In summary, if there is a network interface defined (for example, *eth0*), a link is detected with a link partner, the send and receive packet counters are incrementing, and no error counters are incrementing in the MAC statistics, then there is not an issue with basic send and receive of Ethernet packets.

One common question is if applications such as `ping` or `iperf3` (formerly `iperf`) are needed to do basic send and receive of Ethernet frames. Once Linux has booted, there is an attempt to acquire an IP address using broadcast messages. The traffic from these broadcast messages are what is used to do basic send and receive of Ethernet frames. Find more details on how IPv4 addresses are obtained in Section 8.

Note

Debug Suggestions:

- If there is no transmitted traffic, several reasons can cause this such as no PHY clocking, PinMuxing, and so forth. See also Section 6.
- If there is no received traffic, there can be one of several reasons such as clocking, data timing, PinMuxing, and so forth. See also Section 6 and review the relevant DTS properties.
- Review *Ethernet MAC Statistics Analysis*.

Section Summary:

- Use `ethtool <interface name>` to check for a link in the desired interface. If there is no link then this issue must be debugged. No data can pass without a link detection.
- If a link is detected then `ethtool -S <interface name>` can be used to detect basic Ethernet frame (or packet) send and receive (that is, to view the Ethernet MAC hardware statistics). If there are Ethernet frames or packets detected, then the interface is working at a basic level. If the statistics indicate no Ethernet frames or packets have been sent or received, then this is the area to debug.

4 Debug Overview

Section 3 has a discussion on some initial debug checks to determine the status of basic *send and receive* of packets. A technique called *follow-the-packet* is introduced here which uses the Ethernet MAC interface statistics to follow the path that an Ethernet packet goes through for transmission and reception at the wire level. Figure 4-1 outlines the basic requirements to send and receive packets. The rest of this application note demonstrates how to analyze each stage of the block diagram.

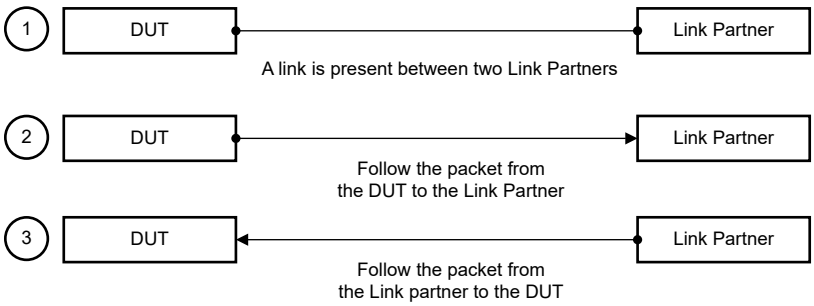


Figure 4-1. Follow the Packet Flow

Fundamental steps of packet flow between two link partners:

1. Packets can flow only after an Ethernet link is established between the two link partners. As part of establishing the link, auto-negotiation is typically used to determine the best bit line rate and duplex mode. Remember, auto-negotiation is triggered upon the physical connection between Ethernet PHYs. In addition to link establishment, the Linux drivers must have initialized correctly for the Ethernet MAC, the MDIO interface and the respective Ethernet PHY. See also *Network Driver Initialization Process*.
2. After a link is established, the transmit path can be examined independently. This application note shows how to make sure that packets were sent on the wire and received by the link partner at the MAC level using MAC hardware statistics. It does not matter at this level what protocols are used, only that Ethernet packet were transmitted.
3. Like the transmit path, the receive path can be examined independently. This application note shows how packets are received at the MAC level by viewing the MAC hardware statistics at the DUT Ethernet interface receiving the packets from the link partner.

Figure 4-2 shows the connection between the Ethernet MAC and the Ethernet PHY. It is assumed here that the reader knows the function and purpose of the Ethernet MAC and the Ethernet PHY. The Ethernet MAC handles the sending and receiving of digital data and the PHY handles the converting of Ethernet Packets between the digital and analog domains.

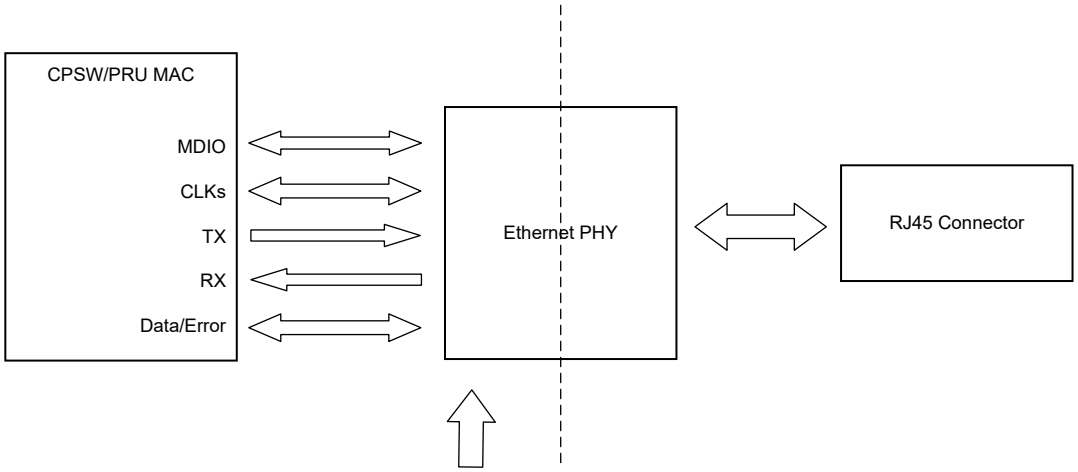


Figure 4-2. Ethernet MAC and Ethernet PHY Interface

The components in Figure 4-2 show the data path (RX, TX), the control path (MDIO), and the error path or synchronization path between the Ethernet MAC and the Ethernet PHY. These are the most general signals needed between a MAC and PHY interface. Depending on the interface mode (MII, RGMII, RGMII, and so forth) between the MAC and PHY, there are different variations of these signals.

5 Network Driver Initialization Process

On TI Arm® based processors there are two drivers that initialize during a Linux boot: the MDIO driver and the CPSW or PRU_ICSSG (or both) driver. The MDIO driver is used for communicating with the Ethernet PHY. The CPSW or PRU_ICSSG (or both) driver set up the Ethernet MAC. To check if these drivers are initialized properly, the boot log of the DUT needs to be examined. The following initialization stages show up:

1. Both the MDIO and CPSW or PRU_ICSSG drivers have initialized without errors.
2. The MDIO driver has detected the PHYs at the respective PHY address defined in the board DTS file. The exact PHY address is determined by the PHY hardware boot strapping.
3. The Ethernet interface is up and a link is established, assuming a cable is attached to the Ethernet port before boot up.

The following log shows an example of what is expected as part of the initialization. The timestamps printed out for each line can differ between different boot sequences.

```
[    1.060850] davinci_mdio 8000f00.mdio: Configuring MDIO in manual mode
[    1.095345] davinci_mdio 8000f00.mdio: davinci mdio revision 9.7, bus freq 1000000
[    1.098179] davinci_mdio 8000f00.mdio: phy[0]: device 8000f00.mdio:00, driver TI DP83867
[    1.098198] davinci_mdio 8000f00.mdio: phy[1]: device 8000f00.mdio:01, driver TI DP83867
[    1.098246] am65-cpsw-nuss 8000000.ethernet: initializing am65 cpsw nuss version 0x6BA01103,
cpsw version 0x6BA81103 Ports: 3 quirks:00000006
[    1.098500] am65-cpsw-nuss 8000000.ethernet: initialized cpsw ale version 1.5
[    1.098505] am65-cpsw-nuss 8000000.ethernet: ALE Table size 512
[    1.099141] pps pps0: new PPS source ptp0
[    1.099456] am65-cpsw-nuss 8000000.ethernet: CPTS ver 0x4e8a010c, freq:500000000, add_val:1 pps:1
[    1.120288] am65-cpsw-nuss 8000000.ethernet: set new flow-id-base 19
```

In this example, the MDIO driver is the `davinci_mdio` and the CPSW driver is `am65-cpsw-nuss`. There are two TI DP83867 PHYs on the DUT and the expected PHY addresses are boot strapped to address 0 and 1. The MDIO driver detected these PHYs at these addresses as specified in the board DTS and attached them to the DP83867 PHY driver. No Ethernet cable was connected before the boot up sequence so a link up message does not show up.

The following log shows the MDIO driver scanning the bus for the Ethernet PHYs that have been defined in the DTS file for the DUT. This is an expected normal initialization sequence. The PHY vendor is identified and the corresponding PHY driver is identified as well.

```
[    6.902614] am65-cpsw-nuss 8000000.ethernet eth1: PHY [8000f00.mdio:01] driver [TI DP83867]
(irq=POLL)
[    6.902648] am65-cpsw-nuss 8000000.ethernet eth1: configuring for phy/rgmii-rxid link mode
[    6.938874] am65-cpsw-nuss 8000000.ethernet eth0: PHY [8000f00.mdio:00] driver [TI DP83867]
(irq=POLL)
[    6.938911] am65-cpsw-nuss 8000000.ethernet eth0: configuring for phy/rgmii-rxid link mode
```

The PHY is defined in the board DTS file is then referenced by the CPSW port in the respective DTS definition. The following code is an edited portion from a TI EVM showing the MDIO with the PHY for an Ethernet port and how the MDIO and PHY are identified in the Linux console during a Linux boot.

```
A DTS PHY node example, shortened for discussion purposes

&cpsw3g_mdio {
    cpsw3g_phy0: ethernet-phy@0 {
        reg = <0>;    <-- check PHY address, verify this is correct
        ......
    };
};

This line shows the PHY defined in the DTS being identified

[    6.938874] am65-cpsw-nuss 8000000.ethernet eth0: PHY [8000f00.mdio:00] driver [TI DP83867]
(irq=POLL)
```

If the MDIO driver is not able to communicate with an Ethernet PHY, there are several errors that can occur that are either related to the DTS or problems in the PHY state from a hardware implementation issue.

Note

Debug suggestions for PHY initialization issues:

- Confirm the PHY node in the DTS is correct
- See also *Ethernet PHY Analysis*

Now that the Ethernet PHYs are identified, these PHYs can be interrogated to see if a link is established with a link partner. If an Ethernet cable is connected and a link is established, the following logs appear on the DUT console. Data bit rate and duplexity are agreed upon when a link is established. Notice in this example, the speed is 1Gbps, the link is full duplex, and the Ethernet interface is ready to use.

```
[   11.042192] am65-cpsw-nuss 8000000.ethernet eth0: Link is Up - 1Gbps/Full - flow control rx/tx
[   11.042255] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
```

The Ethernet speed and duplex mode on the TI EVMs are agreed between link partners using auto-negotiation. The above message shows a successful initialization sequence.

If there are issues with the driver initialization, the recommendation is to review the board DTS with the TI EVM that the board is designed around. The following code is a sample of the DTS file that is used to define the CPSW interface (Ethernet MAC) for the TI SK-AM62B-P1 EVM that can be used as a reference.

The following DTS example shows some checkpoints that need to be verified when working with a custom board DTS.

```
Linux Source tree directory: arch/arm64/boot/dts/ti/k3-am62x-sk-common.dtsi
...
&cpsw3g {
    bootph-all;
    pinctrl-names = "default";
    pinctrl-0 = <&main_rgmii1_pins_default>;   <-- verify pin mux is correct
};

&cpsw_port1 {
    bootph-all;
    phy-mode = "rgmii-rxid";         <-- verify interface mode is correct
    phy-handle = <&cpsw3g_phy0>;     <-- verify phy name is referenced correctly
};
...
&cpsw3g_mdio {
    bootph-all;
    status = "okay";
    pinctrl-names = "default";
    pinctrl-0 = <&main_mdio1_pins_default>;    <-- verify pin mux is correct

    cpsw3g_phy0: ethernet-phy@0 {
        bootph-all;
        reg = <0>;
        ti,rx-internal-delay = <DP83867_RGMIIDCTL_2_00_NS>; <-- rx clock delay
        ti,fifo-depth = <DP83867_PHYCR_FIFO_DEPTH_4_B_NIB>;
        ti,min-output-impedance;
    };
};
```

Section Summary:

The following are the conditions that indicate a successful initialization process:

- The MDIO driver has initialized without errors and has identified all expected PHYs on the MDIO bus.
- The CPSW or PRU_ICSSG (or both) driver has initialized without errors.
- The CPSW driver is reporting a link up message with the expected bit rate and duplexity.

6 Ethernet PHY Analysis

This section covers some basic Ethernet PHY debug steps to make sure that the PHY has initialized and is operating correctly.

The first area to examine are the power supply and clock out signals of the Ethernet PHY. If possible, look for a troubleshooting guide from the Ethernet PHY vendor. For example, the DP83869 Ethernet PHY from TI has a troubleshooting guide that demonstrates how to do basic evaluation of the DP83869 PHY.

For discussion purposes, the *DP83869 Troubleshooting Guide* is used as an example on basic Ethernet PHY voltage and clocking debug.

There are some peripheral pins on the Ethernet PHY to pay close attention to. As a reference, these pins are specified in the *peripheral pin checks* section of the referenced troubleshooting guide. The referenced troubleshooting guide contains significantly more details.

- Power Supplies
  - Verify each voltage rail is within tolerance specified by the Ethernet PHY data sheet of the manufacturer
- Probe the XI Clock
  - The input clock is necessary for the PHY to be operational
  - Confirm the signal is present and meets the specifications of the PHY
- Probe the RESET_N Signal
  - Verify the PHY is not being held in reset
- Probe the Strap Pins During Initialization
  - This is how the PHY finds the configuration
    - Example: the PHY address on the MDIO bus is one of the configurations
  - The troubleshooting guide suggests a specific debug technique to follow

  - See also, the *How to Confirm Ethernet PHY Strapping* TI E2E™ FAQ
- Probe the Serial Management Interface Signal (MDC, MDIO)
  - Reference the expected resistive tolerances and signal integrity requirements from the PHY data sheet
  - This step is not necessary if the PHY registers are accessible through the `ethtool` utility (for example, `ethtool eth0`)

If the peripheral pins are verified to be correct and the MDIO bus is correctly signaling; however, communication is not happening with the PHY, the next debug step is to check the MDIO driver and the selected PHY driver. Analyzing the MDIO driver and communication is covered in Section 5. An additional reference is in the `dmesg | grep -i mdio` section of the *How to Integrate Linux Driver* application note which covers an MDIO interaction example.

Pay particular attention to the MAC - PHY interface when in RGMII mode. The key debug element here is that RX clock from the PHY to the MAC needs to be delayed relative to the RX data. If this timing is not correct, this causes issues with the reception of Ethernet frames. Errors such as dropped frames or RX CRC errors are measured at the MAC. TI Arm® based processor MACs automatically apply a 2ns delay on the TX clock relative to the TX data when operating in the RGMII interface mode. An additional note is that board layout and trace length of the clock and data lines are critical to defining the correct clock delay. The *High-Speed Interface Layout Guidelines* application note provides information on the importance of high-speed layout.

The board DTS defines the RX clocking delay information to be used by the PHY driver. See the *How to Integrate Linux Driver* application note for an example on how the RX clocking delay is configured.

The following snippet is a DTS example of how the RX clocking delay is configured.

```
(Linux Source tree directory) arch/arm64/boot/dts/ti/k3-am62x-sk-common.dtsi (a portion from this
file)

#include <dt-bindings/net/ti-dp83867.h> <-- Has the macro defines used in the phy node

cpsw3g_phy0: ethernet-phy@0 {
bootph-all;
    reg = <0>;
    ti,rx-internal-delay = <DP83867_RGMIIDCTL_2_00_NS>; <-- rx clock delay
    ti,fifo-depth = <DP83867_PHYCR_FIFO_DEPTH_4_B_NIB>;
    ti,min-output-impedance;
};
```

An example of the required RX clock delay here is referenced from the DP83869 Ethernet PHY data sheet that defines the necessary RX clock delay.

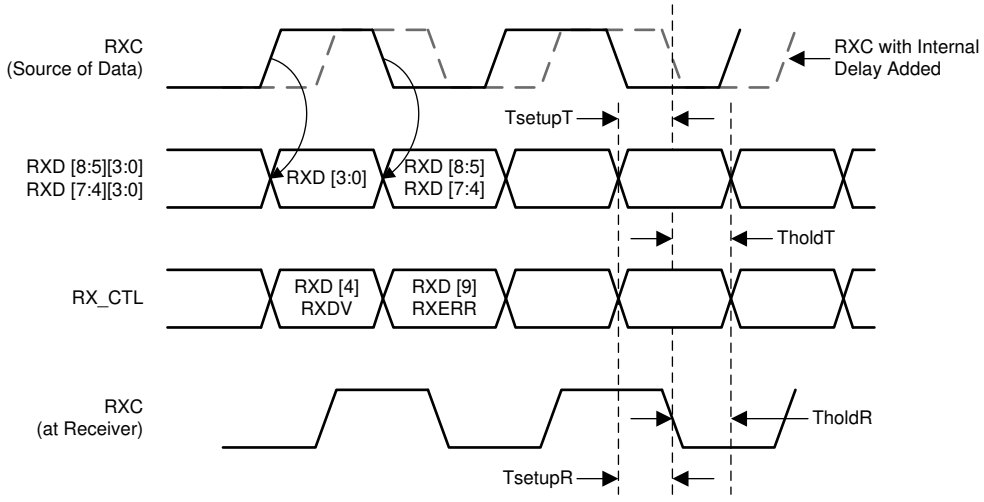


Figure 6-1. RGMII Receive Multiplexing and Timing Diagram

Section Summary:

How to verify if the Ethernet PHY is operating

- Verify the peripheral pins are either operating as expected and within tolerances specified by the PHY data sheet and the troubleshooting guide for the PHY if one exists
- Review that the MDIO driver has initialized without errors and has identified all the expected PHYs in the Linux console log.
- If the PHY interface is RGMII, check that the RX clock is delayed per the board layout and verify the RX clock delay is configured correctly in the board DTS file.

7 Ethernet MAC Statistics Analysis

Review the MAC statistics for errors between the DUT and the link partner to verify that basic transmission and reception of Ethernet packets was successful. Remember that even if basic transmission and reception of packets occur, there can be errors that are limiting network throughput. Some errors and solutions are discussed in this section.

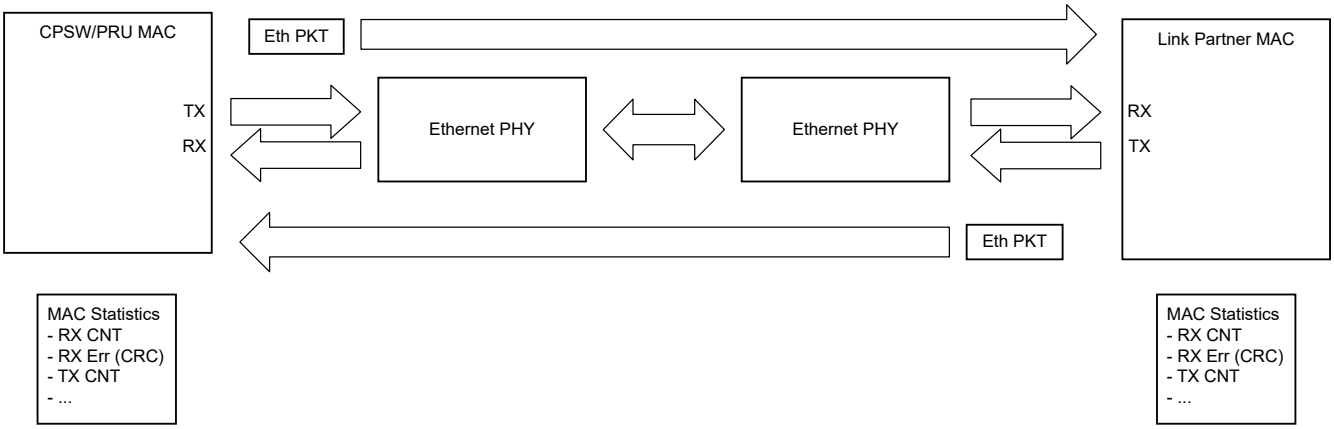


Figure 7-1. Full Ethernet Packet Flow With MAC Statistics

Reviewing the MAC statistics indicates the number of packets sent, received, and the number of ones dropped with errors. Remember that MAC statistics are the results from monitoring the packets directly at the interface.

Analyzing the granularity of an individual packet when examining the interface statistics is not necessary. The main behavior to look out for are incrementing counts for RX and TX good frames and checking that there are no errors detected such as CRC, alignment, and so forth.

Incrementing TX and RX good frames statistics registers are an indication that sending and receiving packets between the MAC and the PHY is working as expected. This is also an indication that the MAC-PHY TX and RX clock is working correctly.

If the MAC statistics error counters are increasing, then there are packets being dropped at the interface. For example, a non-zero RX CRC error counter indicates that packets are received with CRC checksum errors and, as a result, dropped at the interface.

Note

TX errors such as CRC errors are not among the list of MAC statistics on the platform under test that can be captured. TX CRC errors are only detected by the RX error counters of the link partner.

The following MAC statistics are in the context of a TI EVM directly connected to another board by connecting an Ethernet cable between the two link partners. Neither link of the two link partners are running a DHCP server to dynamically assign IP addresses. As Section 3 mentioned, you do not need to obtain an IP address to examine packet traffic. When examining traffic at this level, the key takeaway is to determine that no errors are observed during basic packet send and receive.

Once a link up on an interface is detected and the MAC driver is initialized, there is network traffic initiated in an attempt to acquire an IP address. The following TX and RX are the result of this network traffic, indicating how many packets were transmitted and received.

Specifically, the example snippet of the MAC statistic for TX packets is a result of connecting an Ethernet cable in between the two link partners. Once the link is detected to be up on an interface, the Linux kernel attempts to acquire an IP address. Some of the frames reported on the *tx_broadcast_frames* statistics are the DHCP request frames from the DUT that are used to acquire an IPv4 address. Furthermore, if the *tx_good_frames* statistic are incrementing, then this is a good an indication that packets were transmitted from the MAC of the DUT onto the wire.

```
A portion of an example ethool -S <interface> output showing the TX MAC statistics:

tx_good_frames: 19
tx_broadcast_frames: 2
tx_multicast_frames: 17
```

The following RX results from the MAC statistics reveal that packets are received as well after connecting an Ethernet cable. These particular statistics show that there are packets correctly received by the MAC. With this example, there is not an IP address assigned to the platform. The actual Ethernet protocol in use does not matter at this level. The main goal is to check that no corrupted packets are received and that packets are properly sent.

```
A portion of an example ethtool -S <interface name> output showing RX MAC statistics:

rx_good_frames: 104
rx_broadcast_frames: 10
rx_multicast_frames: 94
```

The next example gives a closer view on the RX error results from the MAC statistics. Here, there are no errors detected. However, if these error statistics are incrementing, it gives an indication to what particular errors caused the packets to be dropped at the interface. For Texas Instruments' processors serving as the DUT, the definitions of each MAC statistics counter can be found in the respective Technical Reference Manual (TRM) for that particular processor.

```
Portion of sample ethtool -S <interface name> showing RX error statistics:

rx_crc_errors: 0
rx_align_code_errors: 0
rx_oversized_frames: 0
rx_jabber_frames: 0
rx_undersized_frames: 0
```

Note

Debug Suggestions:

- If the `ethtool` utility tool is not returning any data for either link detection or MAC statistics, then this indicates the MDIO communication to the PHY is not performing as expected. Confirm the PHY device driver was loaded and correctly identified in the Linux boot log. If the PHY was not correctly identified, consult Section 6.
- If there is TX traffic, but no RX traffic, then see Section 4, Section 9, or the debug suggestions at the end of each section in this application note.

Section Summary:

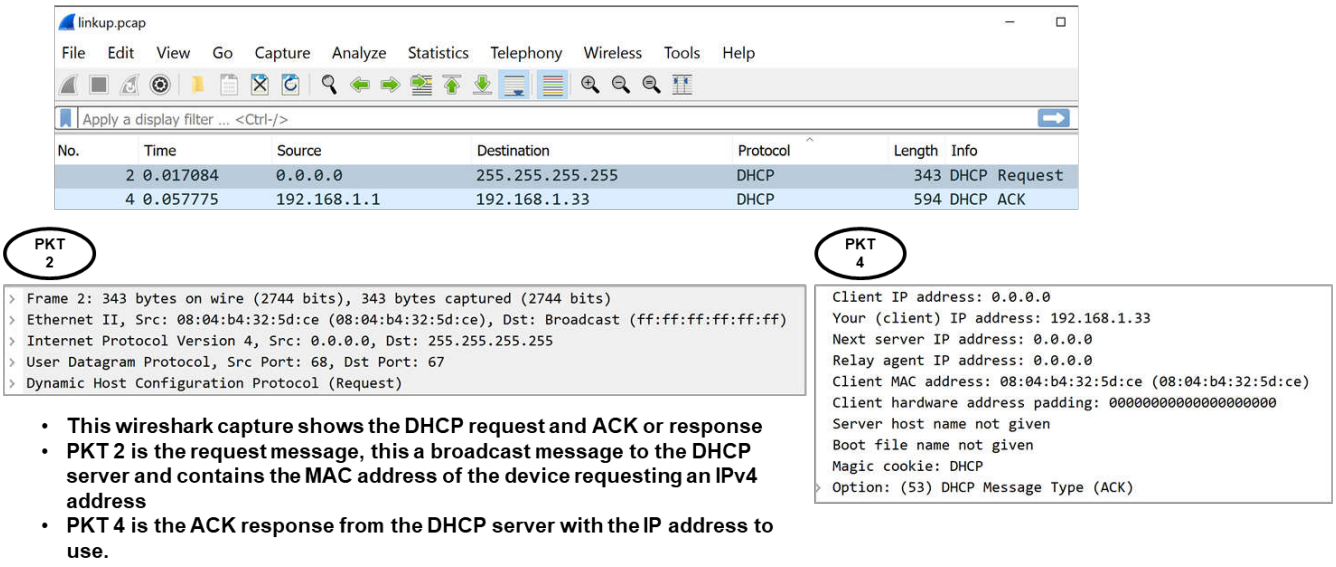
- The results of `ethtool -S <interface name>` for an interface provides data on the send and receive process.
- Connecting the Ethernet cable between two link partners generates traffic that can be viewed with `ethtool` without the need for obtaining an IP address.

8 How IPv4 Address is Obtained

While successfully getting an IPv4 address assigned to the DUT Ethernet interface under test is not required, the process of obtaining an IPv4 address provides a means to test basic transmission and reception of packets. On a default TI SDK filesystem, an attempt is made to acquire an IP address when an Ethernet link is detected to be up. Most users assume there is an issue with the network connection when an IPv4 address is not received. However, this is the case if a DHCP server is setup in the network topology. If no DHCP server is setup, not receiving an IPv4 address is not always an indication that the network connection failed, just that no server was set up to assign the IPv4 addresses.

The first step to obtaining an IPv4 address is to make sure the DUT is connected to a link partner that either has a DHCP server or to a network that has one. If there is no DHCP server, then this IP address acquisition fails and gives the appearance that the platform under test is not sending or receiving packets.

Figure 8-1 shows that after the network driver has initialized, the network manager sends a DHCP request for an IP address. The DHCP server provides a response or ACK that contains the IP address that the DUT is assigned to.



- **This wireshark capture shows the DHCP request and ACK or response**
- **PKT 2 is the request message, this a broadcast message to the DHCP server and contains the MAC address of the device requesting an IPv4 address**
- **PKT 4 is the ACK response from the DHCP server with the IP address to use.**

Figure 8-1. Wireshark DHCP Messages

If this IP address acquisition process does not successfully complete, the next step is to follow the path of the packet from the DUT transmitting the packet to the link partner receiving the packet. This *follow-the-packet* (FTP) process also involves the link partner response from transmitting to the DUT receiving the transmitted packet. Section 9 describes the analysis of this FTP process.

9 Follow the Packet

Once an Ethernet link is established between two link partners, one can trace the steps of the packet sent from the DUT to the link partner and then packets from the link partner back to the DUT. This only works if you have access to the Ethernet MAC statistics on both sides of the link. This is also made simpler if the network topology consists of a direct connection shown by Figure 3-1.

As described in Section 8, you do not need to set IP addresses to initiate network traffic. When using the TI default filesystem on the DUT, there is an attempt to acquire IP addresses after a link is detected. This network communication is sufficient to trace successful reception of transmitted packets from the DUT to the link partner. Broadcast packets from the link partner are sufficient to prove that no corrupted packets were received on the DUT.

Before connecting an Ethernet cable between the DUT and the link partner, use `ethtool` to verify that the Ethernet MAC statistics are 0 for the RX and TX counters. The recommendation is to use a Linux® operating system (OS) on the link partner platform. This can be another working EVM or an Ubuntu PC. `ethtool` is only available for platforms running a Linux OS.

As Figure 9-1 shows, start with examining if packets sent from the DUT are received by the link partner by using `ethtool` on the link partner and checking the RX counters. If the link partner is not running on a Linux OS, `ethtool` is not available. The RX counters give an indication to how many good frames are received and more importantly, how many corrupted frames are received. If the packets received are error free, then the transmit path from the DUT is working as expected. If the RX error counters on the link partner are nonzero (for example, RX CRC errors), then a conclusion can be made that there is an issue to investigate for the transmit path of the DUT.

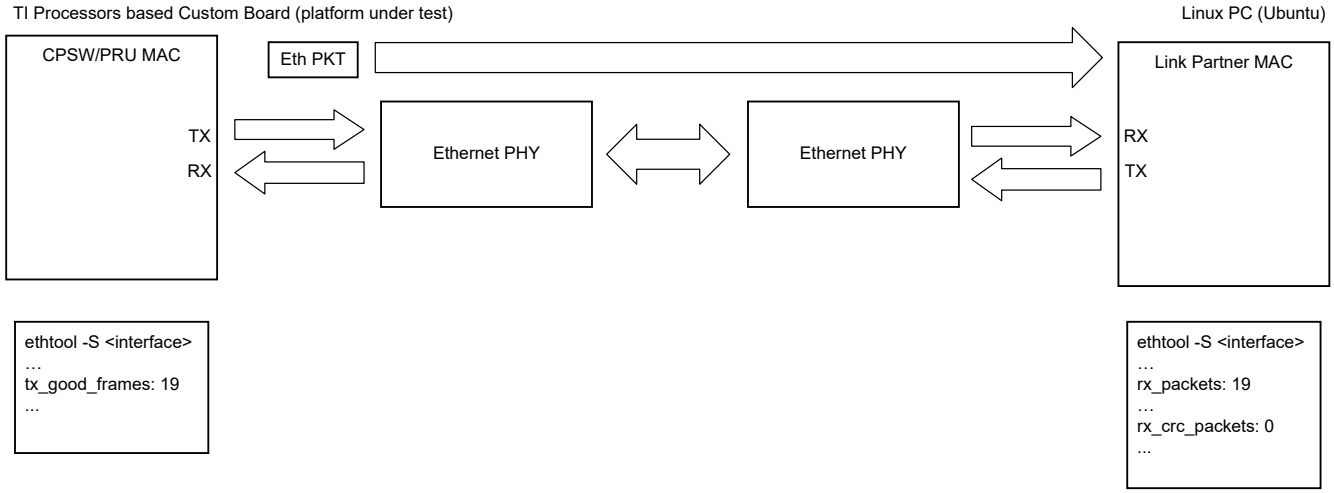


Figure 9-1. DUT Transmit Path

There are typically two issues that can be observed by a receiving link partner: no packets received or CRC errors. If the link partner is not receiving any packets, then something is corrupting the Ethernet frame from the DUT to the point of the frame not being recognized by a MAC interface. When either no packets or CRC errors are detected, the MAC or PHY interfacing on the DUT needs to be examined more closely.

Note

Debug Suggestions:

- When the issue is that no packets are being received by the link partner
  - Review and verify the PinMux in the DTS file
  - Verify the PHY interface mode (for example, RMII or RGMII) in the DTS to the schematic
- If the link is receiving CRC or other errors:
  - Try changing the Ethernet cable in case the cable was damaged
  - Try a second known good link partner, that has access to the Ethernet MAC statistics
  - If using a PHY is RGMII mode, and no RX clock delay was added by the PHY, add this delay. TI Arm® based processors already set a 2ns TX clock delay so the PHY does not add any delay to the TX clock.
  - Review Section 6 for details on how to analyze transmit errors

Analyzing the receive path to the DUT is similar to how the transmit path from the DUT is analyzed. Observe in Figure 9-2, that the receive packet count and some errors that can happen. As indicated previously, a receive path is functioning correctly when the error counters are zero and the received good frame count is increasing. At this level, it does not matter the types of packets, just that the packets are being received and no error packets are being detected.

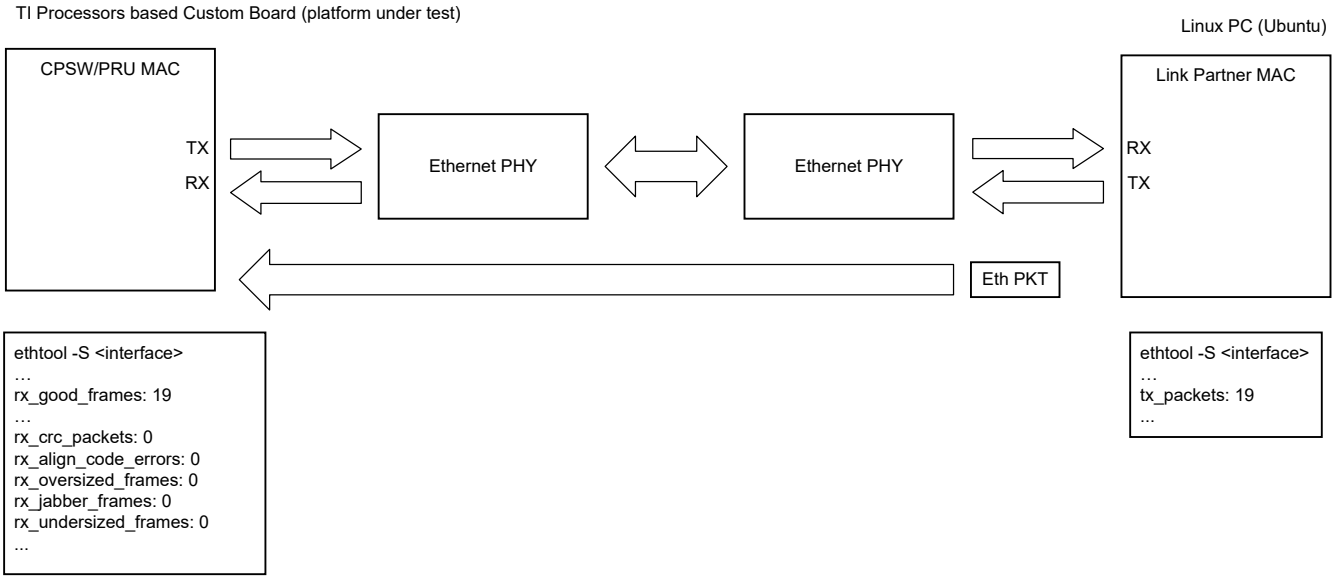


Figure 9-2. DUT Receive Path

There are usually two possible causes of when the Ethernet MAC receive error counters are nonzero. The most common is due to an incorrect RX clock timing, when the PHY interface mode is RGMII. The second most common issue is a hardware issue such as incorrect in-line resistor values or a board layout issue.

Note

Debug Suggestions:

- If the PHY interface mode is RGMII, then review if the layout timing matches what was programmed in the PHY node of the DTS file.
- Review the board layout using the *High-Speed Interface Layout Guidelines* application note.

If the error counters from analyzing the DUT transmission and reception paths are zero, then basic Ethernet packets are being sent and received error free. At this point, a DHCP server can be added to assign IP addresses to the DUT dynamically or an IP address can be assigned statically with `ip addr add <ip address/subnet mask> dev <interface name>` on the DUT. However, after this step is performed, if

there is still no IP address assigned when the link is detected as up and the Ethernet MAC statistics are not showing any errors, then the next step is to consider possible network reasons. Review Section 10 for examples of network issues.

Section Summary:

- This section shows how to analyze basic Ethernet packet transmission and reception using the FTP method.
- If packets are error free and no IP address is obtained, then this can allude to a network-level issue.

10 Debug Network Topologies and Techniques

The debug techniques described in this section can be helpful under different network issues.

10.1 Direct Connection

The simplest topology is to directly connect two link partners together without a switch or WLAN router. A direct connection example is between the DUT and a known good test platform such as an Ubuntu Linux PC. Figure 10-1 shows a direct connection where there is not a switch or WLAN router connecting the two link partners.

Connecting the two link partners generates traffic without further action because both platforms send out broadcast packets in an attempt to acquire an IP address.

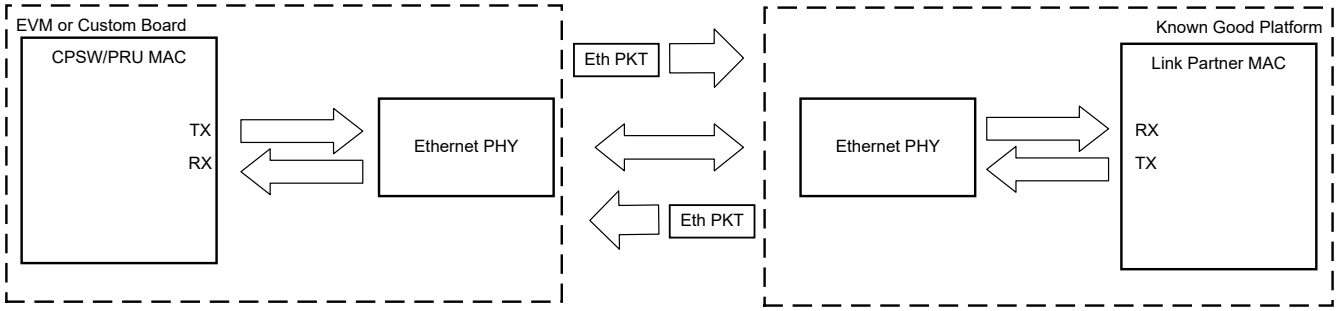


Figure 10-1. Direct Connection Topology

10.2 Static IP Addresses

To initiate traffic using utility tools such as `ping` or `iperf`, set an IPv4 address. This can be done with a DHCP server to assign dynamic IP addresses or done manually with static IP addresses. When using the static method and a direct connection topology, both link partners need an IP address manually assigned to the same subnet. For example, if the IPv4 address is 192.168.1.1 with a netmask of 24, then the first three set of numbers (that is, 192.168.1.x) is the subnet. Section 11 has an example of how to set static IP addresses at runtime.

10.3 Setting Low Bit Rates

When using the RGMII interface mode, timing issues related to clock delay with respect to data can be determined by using a method of lowering the maximum bit rate allowed for the Ethernet link. This limit can be done with three methods:

1. Use `ethtool` to manually set the link speed on both link partners:
   `ethtool -s <interface name> speed <link speed>`
2. Connect to a lower bit rate link partner
3. Add a *max-speed* property to the DTS to lower the bit rate. This method is preferred as this is set on each DUT during boot time. The following code snippet shows an example of adding this property to the PHY node in a DTS file. Observe that the speed is set to the lowest setting of 10 indicating 10Mbps. 100Mbps can also work. A default speed of 1000Mbps from the PHY HW configuration is assumed here.

```
cpsw3g_phy1: ethernet-phy1@ {
   …
   max-speed = <10>;
   …
};
```

10.4 Beware When Connecting to a Switch

Do not connect both Ethernet interfaces on a dual port board, such as a TI EVM into the same switch or router. This does not provide a redundant link. Each Ethernet interface must have an exclusive subnet.

To explain why this network topology fails, assume two interfaces on the DUT are named *eth0* and *eth1* and are both connected to the same switch. When a `ping` command (issuing an ICMP request) is initiated from *eth1* (`ping <Ipv4 address> -I eth1`), the packet that is sent out by the DUT is received by the destination platform. To respond to the ping request, the destination platform must perform an Address Resolution Protocol (ARP) sequence to get the MAC address of the source platform that made the ping request. This ARP sequence is called an ARP request message.

Since the ARP packet is a broadcast message, *eth0* of the source platform responds with the MAC address, which is incorrect for *eth1*.

If the ARP broadcast returned the wrong MAC address for *eth1*, every ping reply from the destination platform has the incorrect destination MAC address. Therefore, *eth0* is receiving all the responses instead of *eth1*. If Wireshark is used to view the packet traffic, it looks like every `ping` request has a matching `ping` reply except that none of the response packets are received by *eth1*, causing a `ping` failure.

For example, in Figure 10-2, the highlighted ping reply is to a destination MAC address of *d4:f5:13:87:0b:8c*, which is the MAC address associated with eth0.

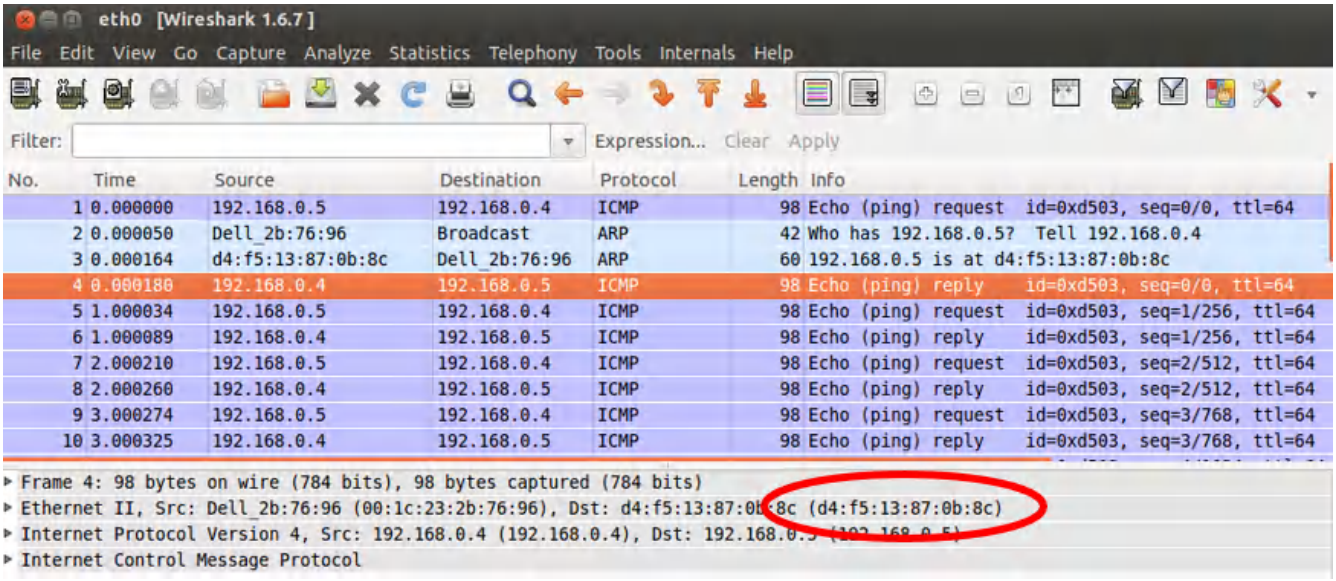


Figure 10-2. Wireshark Ping Messages

11 Linux® Utilities Summary

This section describes the Linux utility tools that are used to perform control and status operations. There are three commonly used utilities in Ethernet interface debugging: `ifconfig`, `ip`, and `ethtool`. The following are brief descriptions about each utility.

- `ifconfig` - Provides basic information regarding the interface. The output provides a summary of the current state of the interface. The following example shows that the interface is up and has an IP address (192.168.1.5). There are also counters for the number of RX and TX packets. The example given here shows a normal working interface. The `ifconfig` utility only gives statistics on a higher networking layer than `ethtool`, which gives the data-link layer results. Additionally, `ifconfig` is not as up to date as the `ip` utility tool.

```
root@am62xx-evm:~# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.5  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::1e63:49ff:fe1f:d9b2  prefixlen 64  scopeid 0x20<link>
        ether 1c:63:49:1f:d9:b2  txqueuelen 1000  (Ethernet)
        RX packets 552490  bytes 61792471 (58.9 MiB)
        RX errors 0  dropped 6  overruns 0  frame 0
        TX packets 646  bytes 50358 (49.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

- `ip` - Networking tool for controlling and interrogating the network interfaces. The `ip` utility is more versatile than the `ifconfig` utility.
  - The following example shows how to list the IP address for the *eth0* interface.

```
root@am62xx-evm:~# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 1c:63:49:1f:d9:b2 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.5/24 metric 1024 brd 192.168.1.255 scope global dynamic eth0
       valid_lft 73177sec preferred_lft 73177sec
    inet6 fe80::1e63:49ff:fe1f:d9b2/64 scope link
       valid_lft forever preferred_lft forever
```

  - The following example shows how to statically set an IP address with the `ip` utility. This is used in a direct connect topology when there is not a DHCP server present.

```
root@am62xxsip-evm:~# ip addr add 192.16.1.1/24 dev eth0
root@am62xxsip-evm:~#
root@am62xxsip-evm:~# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 08:04:b4:32:5d:ce brd ff:ff:ff:ff:ff:ff
    inet 192.16.1.1/24 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::a04:b4ff:fe32:5dce/64 scope link
       valid_lft forever preferred_lft forever
```

  - The following example shows how to check the RX and TX counters with the `ip` utility.

```
root@am64xx-evm:~# ip -s link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group
default qlen0
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    RX:  bytes packets errors dropped  missed   mcast
          9246      99      0       0       0       0
    TX:  bytes packets errors dropped carrier collsns
          9246      99      0       0       0       0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group
default ql0
    link/ether 1c:63:49:1a:da:62 brd ff:ff:ff:ff:ff:ff
    RX:  bytes packets errors dropped  missed   mcast
         19654      61      0       0       0       0
    TX:  bytes packets errors dropped carrier collsns
         22919      88      0       0       0       0
...
```

- `ethtool` - Provides a method for displaying and modifying the parameters of the Ethernet interface and accompanying device driver. There are several options for running `ethtool`. Two of the most commonly used are for examining the link status and the MAC hardware statistics. The specific options for these are given as examples throughout this application note.

12 Checklist for Requesting Ethernet Support

When posting an inquiry on the TI E2E™ forum (related to an Ethernet communication issue using a Linux OS on the DUT), answer the following questions or complete the requested steps in the initial post:

1. What is the TI SDK version in use?
2. What is the TI processor in use?
3. Is the DUT a TI EVM or a custom board?
4. What is the test topology?
5. If using a custom board, attach the DTS file for the custom board; do not cut and paste the DTS contents as the DTS file can be very long creating difficulty for readers to parse
6. Attach the console boot log of the DUT
7. Attach the results of `ethtool <interface name>` and `ethtool -S <interface name>` for the interface with the issue
8. Indicate that you have reviewed this application note

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265