TERIDIAN SEMICONDUCTOR CORP.

Simplifying System Integration™

78Q8430 ARM9(920T) Linux Driver Diagnostic Guide

October 17, 2008
Rev. 1.1
UG_8430_003

Teridian Semiconductor Corp., 6440 Oak Canyon, Suite 100, Irvine, CA 92618
TEL (714) 508-8800, FAX (714) 508-8877, http://www.teridian.com

Table of Contents

1 Introduction

The Teridian Semiconductor Corporation (TSC) 78Q8430 is a single chip 10/100 Ethernet MAC and PHY controller supporting multi-media offload. The device is optimized to enhance throughput and offload network protocol tasks from the host processor for demanding multi-media applications found in Set Top Boxes, IP video, and Broadband Media Appliances.

This document describes the 78Q8430 Linux® software device driver diagnostic procedures. The document is based on the following driver software version:

- 78Q8430 ARM Linux 2.6.13 driver, version 1.1.

The procedures utilize the 78Q8430 Embest Evaluation Board (daughter card) on the Embest Samsung S3CEB2410 ARM9™ platform running Linux kernel 2.6.13. This kernel supports Embest's implementation of Mizi Inc.'s Linuette Linux distribution.

The 78Q8430 Embest Evaluation Board is a 32-bit implementation leveraging the 32-bit data bus of the Samsung S3CEB2410 SOC. As a result, the testing procedures described reflect only 32-bit access to the 78Q8430.

Use this document with those listed in the Related Documentation section.

2 System Requirements

2.1 Hardware Requirements

The following list describes the minimum hardware requirements for the 78Q8430 ARM9 Linux based evaluation and testing platform:

- 78Q8430 Embest Evaluation Board (daughter card).
- The Embest ARM9 S3CEB2410 evaluation platform (also called the ARM Board in this document).
- A PC to serve as the Linux host and control the ARM9 S3CEB2410 platform (also called the Linux Host and the Linux PC in this document).

2.2 Software Requirements

The following list describes the minimum software requirements for the procedures described in this document:

- 78Q8430 ARM Linux 2.6.13 driver, version 1.1.
- Linux operating system supporting kernel 2.6.13.

The Linux 2.6.13 device driver support for the 78Q8430 is configured by default to include debug message support. This is required to show test results and facilitate verification. Debug message support is a compile time build option and can be easily compiled out for production operation.

3 Diagnostic Tools

The 78Q8430 10/100 Ethernet MAC/PHY Linux device driver provides extensive configuration support through its IOCTL interface. The 78Q8430 is a uniquely feature rich ethernet solution. As such, the standard Linux ethernet configuration applications, ethtool and mii-tool, do not address the majority of the available features. For this reason, a new configuration application, the 8430-tool, is provided with the driver to support the chip's feature set. The 8430-tool application uses the Linux standard set of sixteen user definable networking IOCTLs. The diagnostic procedures utilize the following utilities:

- 8430-tool: for the chip features that can not be addressed by the standard Linux ethernet command interfaces.
- ethtool: for standard Linux ethernet control commands.
- mii-tool: for standard Linux ethernet MII control commands.

This section contains a summary of the 8430-tool, ethtool and mii-tool commands. Refer to the *78Q8430 Linux Driver ARM Platform User Guide* for a detailed description of the 8430-tool commands.

3.1 ethtool Command Syntax

The following syntax summary lists the ethtool version 4 command options supported by the 78Q8430 device driver. ethtool can be obtained from: http://sourceforge.net/projects/gkernel/.

The EEPROM modification command described below requires a code conveyed by the 'magic' operand. The required code is *8430*.

```
~# ethtool –h                              Displays command syntax (as shown below)

ethtool version 4
Usage:
ethtool DEVNAME                            Display standard information about device
     ethtool -i|--driver DEVNAME           Show driver information
     ethtool -e|--eeprom-dump DEVNAME      Do an EEPROM dump
          [ offset N ]
          [ length N ]
     ethtool -E|--change-eeprom DEVNAME    Change bytes in device EEPROM
          [ magic 8430]
          [ offset N ]
          [ value N ]
```

3.2 mii-tool Command Syntax

The following syntax summary lists the mii-tool version 4 command options supported by the 78Q8430 device driver. mii-tool can be obtained from: http://www.metux.de/pub/mii-tool/mii-tool-1.9.1.1.tar.bz2.

```
~# mii-tool
Usage: mii-tool [-VvRrwl] [-A media,... | -F media] [interface ...]
    -R, --reset                            reset MII to poweron state
    -F, --force=media                      force specified media technology
       media: 100baseT4, 100baseTx-FD, 100baseTx-HD,
              10baseT-FD, 10baseT-HD,
              (to advertise both HD and FD) 100baseTx, 10baseT
~#
```

4 Diagnostic Procedures

The test procedures use the target Embest platform under test (containing the 78Q8430) and a Linux Host PC, which generates the majority of the tests and provides some required verification facilities.

The test procedures are preceded by the Test Setup section describing the preparation needed for using the testing platform. The test procedures are generally grouped by 78Q8430 features. The major test sections are:

- Packet Tests – basic device operation tests.
- MAC Tests – 78Q8430 MAC tests.
- TOE Tests – TCP offload engine frame transfer tests.

4.1 Test Procedure Nomenclature

The test procedure descriptions use the following nomenclature:

- The target platform referred to in the procedures is the Embest Samsung S3CEB2410 ARM Evaluation Board (ARM Board) containing the 78Q8430 Embest Evaluation Board.
- Minicom is a Linux PC serial console program equivalent to Windows® Hyperterm. It is used as the target platform's console for boot-up, all command input and driver diagnostic message output used in the verification steps of the tests.
- All console commands given in the test procedure are shown in **bold** face type.
  For example: On the next prompt, respond: **boot**
- All console messages referenced in the procedures are shown in *italics*.
- Any command that is given in a procedure with one of the following prefixes is assumed to be entered on the target platform console even if no platform is designated:
  - **8430-tool**…
  - **Mii-tool**…
  - **Ethtool**…
- Some commands on the Linux Host require specifying the Linux PC Ethernet port involved in the testing of the 78Q8430 ARM target system. The string, ***ethN***, is to be replaced by the Linux PC port being used for the testing. ***N*** must be replaced with 0, 1, or 2 (shown by running ifconfig) in correspondence with the Linux PC Ethernet interface that is connected to the target platform.

  For example: replace **tcpdump -lnei *ethN*** with **tcpdump -lnei eth1** if the Linux PC Ethernet port connected to the ARM target is port 1.

4.2 Test Setup

1. Connect the ARM Board to a 5V power supply.
2. Connect an Ethernet cable between the Linux Host and the ARM Board.
3. Connect a serial console cable from the Linux Host serial port to the ARM Board serial port.
4. On the Linux Host, set the testing port IP address to 192.168.10.100 (the ARM platform boots with IP address 192.168.10.99).
5. Open a full height xterm or kconsole window on the Linux Host.
6. Start minicom on the Linux Host at the xterm/kconsole command prompt:
   ~# **minicom**
7. Turn on the ARM Board power switch.
8. Boot the ARM Board. Console messages must appear in the minicom window after power on.
   a. Hit the space bar at the first boot prompt
   b. Respond with: **cpu set 200 2**
   c. On the next prompt, respond: **boot**

9. When boot up completes, enter the following at the prompt:
   arp -s 192.168.10.100 00:0E:2E:5B:25:86
10. On the Linux Host, issue the following ping command to verify the connection to the target:
   ping 192.168.10.99

4.3 Packet Tests

This test group drives TCP/IP generated ping packets from the Linux host to the target containing the 78Q8430 board under test, and vice versa.

4.3.1 Packet Transmit

This procedure tests the packet transmit function using Slave DMA for all transmit I/O.

1. Issue the following ping command from the target platform to the Linux host:
   /sbin/ping 192.168.10.100
2. Observe the ARM Board console output.
3. Verify that the Linux Host responds to each packet sent by the target platform.

4.3.2 Packet Receive

This procedure tests the packet receive function using Slave DMA for all receive I/O.

1. Issue the following ping command from the Linux host to the ARM Board:
   /sbin/ping 192.168.10.99
2. Observe the ARM Board console output.
3. Verify that the ARM Board responds to each packet sent by the Linux host.

4.4 MAC Tests

4.4.1 RX Interrupt Delay Timer

This procedure tests the setting of the RX Interrupt Delay Timer.

1. Connect an oscilloscope to the 78Q8430 interrupt pin (J13, refer to the *78Q8430 Embest Evaluation Board User Manual*).
2. Set the scope to display 1.0 V and 50 µs per division.
3. On Linux host enter: **pktgen.sh**
4. Adjust the time scale of the scope so that at least two distinct interrupt assertions can be seen on the scope and are close together.
5. Record the interval between the two interrupt assertions .
6. Adjust the interrupt delay: **8430-tool eth0 -Q irqdelay set *1000***
   The delay value (*1000* is given as an example) must be changed as needed to demonstrate an increase in the interval between interrupts, since different systems will deliver packets at different rates.
7. Capture and record the new interrupt interval from the scope.

4.4.2 Watermark Interrupt

This procedure tests the setting of the buffer high watermark and handling of the watermark interrupt.

1. Enter the command: **8430-tool eth0 -Q highwater get**
2. Enter the command: **8430-tool eth0 -Q highwater set 15**
3. Enter the command: **8430-tool eth0 -Q highwater get**
4. Verify the change.

To restore the default watermark value, enter: **8430-tool eth0 -Q highwater set 20**

4.4.3 Software Reset

The procedure generates and verifies the software reset.

1. Attach and configure a Logic Analyzer to trap access to register 0x154 (TP2 and TP5, refer to the *78Q8430 Embest Evaluation Board User Manual*).
2. Enter the command: **8430-tool eth0 –R**
3. Verify that the MAC Reset bit has been written to the 78Q8430.

4.4.4 EEPROM Access

This procedure tests random EEPROM read and write access.

1. To read the EEPROM, enter: **ethtool -e eth0 offset 0x00 length 10**
2. Verify the read value.
3. Write to the EEPROM: **ethtool -E eth0 magic 8430 offset 0 value 0x05**
4. Read back the value: **ethtool -e eth0 offset 0x00 length 10**
5. Verify that 0x05 was written.
6. Restore the original value: **ethtool -E eth0 magic 8430 offset 0 value 0xFF**

4.4.5 ARC

This procedure tests the ARC uni-cast, multi-cast and promiscuous mode filters.

<u>ARC Uni-cast</u>

1. Enter the command: **8430-tool eth0 -c filter u 2**
2. Enter the command: **8430-tool eth0 -C arc u 2 00:01:02:0D:0E:0F**
3. Enter the command: **8430-tool eth0 -c filter u 2**
4. Verify the returned RMR low order bytes now show *00,01,02,0D,0E,0F*.
5. Enter the command: **8430-tool eth0 -C arc u 2 00:01:02:03:04:05**
6. Enter the command: **8430-tool eth0 -c filter u 2**
7. Verify the returned RMR low order bytes now show *00,01,02,03,04,05*. This shows that they have been restored to their original values.

<u>ARC Multi-cast</u>

1. Enter the command: **8430-tool eth0 -c filter m 2**
2. Enter the command: **8430-tool eth0 -C arc m 2 01:02:03:04:05:06**
3. Enter the command: **8430-tool eth0 -c filter m 2**
4. Verify the returned RMR low order bytes now show *01,02,03,04,05,06*.
5. Enter the command: **8430-tool eth0 -C arc m 2 00:00:00:00:00**
6. Enter the command: **8430-tool eth0 -c filter m 2**
7. Verify the returned RMR low order bytes are *00,00,00,00,00,00*. This shows that the original values have been restored.

4.4.6 Append CRC

1. On the Linux host enter: **ping 192.168.10.99**
2. Verify that the ARM target responds to the pings.
3. Enter the command: **8430-tool eth0 -F crc fix on**
   Note: CRC append and fix are mutually exclusive. CRC append is the startup default.
4. Verify that packets are no longer returned.
5. Enter the command: **8430-tool eth0 -F crc append on**
6. Verify that packets are returned again.

4.4.7 Strip CRC

1. Enter the command: **8430-tool eth0 -r read 0x154**
2. Verify the returned value is *0x0A000052*.
3. Enter the command: **8430-tool eth0 -F strip crc off**
4. Enter the command: **8430-tool eth0 -r read 0x154**
5. Verify the returned value is *0x0A000050.*
6. Enter the command: **8430-tool eth0 -F strip crc on**
7. Enter the command: **8430-tool eth0 -r read 0x154**
8. Verify the returned value is *0x0A000052*

4.4.8 Strip Padding

1. Enter the command: **8430-tool eth0 -r read 0x154**
2. Verify the returned value is *0x0A000052*.
3. Enter the command: **8430-tool eth0 -F strip pad on**
4. Enter the command: **8430-tool eth0 -r read 0x154**
5. Verify the returned value is *0x0A0000D2*.
6. Enter the command: **8430-tool eth0 -F strip pad off**
7. Enter the command: **8430-tool eth0 -r read 0x154**
8. Verify the returned value is *0x0A000052.*

4.4.9 RMON Read

This procedure tests reading the RMON counters.

1. On the Linux host enter: **ping 192.168.10.99**
2. Enter the command: **8430-tool eth0 -m**
3. Verify that at least 100 packets were received and sent.

4.4.10 RMON Clear

This procedure tests clearing of the RMON counters.

1. Enter the command: **8430-tool eth0 -M clear**
2. Enter the command: **8430-tool eth0 -m**
3. Verify that the counters values are 0s.

4.4.11 RMON Rollover

This procedure verifies the generation of the RMON rollover message.

1. On the Linux host, enter: **ping -f 192.168.10.99**
2. Verify that the ARM target responds to the pings.
3. On the ARM target, turn on TOE mode by entering the following command:
   8430-tool eth0 -T on
4. Run continuously for 1 or more hours.
5. On the ARM target, verify that the RMON rollover message is received.

4.4.12 Pause Frame

This procedure tests the specification and sending of the Pause Frame.

1. Enter the command: **8430-tool eth0 -P frame default**
2. Enter the command: **8430-tool eth0 -P threshold *25***
   The value *25* is arbitrary; it is a count of blocks, so any value up to 120 is acceptable.
3. On the Linux host:
   a. Enter: **tcpdump -lnei *ethN***
   b. Enter: **pktgen.sh**
   c. Verify the transmission of the Pause frame in the tcpdump contents.
4. On the Linux host:
   a. Kill **pktgen.sh**
   b. Enter: **ping 192.168.10.99**
   c. Verify the ping response.
5. Enter the command: **8430-tool eth0 -P frame user**
6. Enter the command: **8430-tool eth0 -P macadr (00:0e:2e:5b:25:86)**
7. On the Linux host:
   a. Enter: **tcpdump -lnei *ethN***
   b. Enter: **pktgen.sh**
   c. Verify the transmission of the Pause frame in the tcpdump contents.
8. On the Linux host:
   a. Kill **pktgen.sh**
   b. Enter: **ping 192.168.10.99**
   c. Verify the ping response.

4.4.13 Pause Watermark

This procedure tests that the application can set the Pause Watermark and can thus enable/disable automatic Pause TX.

1. Enter the command: **8430-tool eth0 -P frame user**
2. Enter the command: **8430-tool eth0 -P macadr (00:0e:2e:5b:25:86)**
3. Enter the command: **8430-tool eth0 -P threshold 0**
   This will halt automatic Pause transmission.
4. On the Linux host:
   a. Enter: **tcpdump -lnei *ethN***
   b. Enter: **pktgen.sh**
   c. Verify that no Pause frames are issued.
5. On the Linux host:
   a. Kill **pktgen.sh**
   b. Enter: **ping 192.168.10.99**
   c. Verify the ping response.

6. Enter the command: **8430-tool eth0 -P threshold *25***
   The value *25* is arbitrary; it is a count of blocks, so any value up to 120 is acceptable.
7. On the Linux host:
   a. Enter: **tcpdump -lnei *ethN***
   b. Enter: **pktgen.sh**
   c. Verify that Pause frames are issued.
8. On the Linux host:
   a. Kill **pktgen.sh**
   b. Enter: **ping 192.168.10.99**
   c. Verify the ping response.

If desired, the 78Q8430 and driver may be returned to the default PAUSE frame state by issuing the following command:

8430-tool eth0 -P frame default

4.4.14 Pause TX

The procedure tests the immediate Pause frame transmission.

1. Perform the Pause Frame procedure.
2. Enter the command: **8430-tool eth0 -P send**
3. Verify the Pause frame transmission.

4.4.15 Local Pause

This procedure tests enabling and disabling of the local Pause counter.

1. Enter the command: **8430-tool eth0 -c rule 0x7D**
2. Verify that the PH Mask field is *0xFE (0x7F)*.
3. Enter the command: **8430-tool eth0 -P local off**
4. Enter the command: **8430-tool eth0 -c rule 0x7D**
5. Verify that the PH Mask field is *0x00*.
6. Enter the command: **8430-tool eth0 -P local on**
7. Enter the command: **8430-tool eth0 -c rule 0x7D**
8. Verify that the PH Mask field is *0xFE (0x7F)*.

4.4.16 Magic Packet WOL

1. Enter the command: **8430-tool eth0 -L magic set 00:0E:2E:5B:25:86**
2. Enable 8430 Power management by entering: **8430-tool eth0 -L pmectl on**
3. Set the 8430 into low power mode by entering: **8430-tool eth0 -L pmepwr on**
4. Enable the wakeup interrupt debug message by entering: **8430-tool eth0 -r debug 0x00040000**
5. On the Linux PC, issue the wol command using the specified MAC address:
   wol -v –ipaddr=192.168.10.99 00:0E:2E:5B:25:86
6. Verify that the wake interrupt is received
7. Enter the command: **8430-tool eth0 -L pmepwr off**
8. Enter the command: **8430-tool eth0 -L pmectl off**

4.4.17 OnNow WOL

1. Enter the command:
   8430-tool eth0 -L onnow set 0x10203040 0x50607080 0x90A0B0C0 0x00000CD1
2. Enter the command: **8430-tool eth0 -c dump 0x15 6**
3. Verify that the stored CAM values make up the group: *0x403080705090*.

4.4.18 HNR Frame

1. Enter the command:
   8430-tool eth0 -H set 0x000e2e61 0x91790001 0x02030405 0x08004500 0x001e0009
2. The 8430-tool will now be in interactive mode. A prompt will appear for another 5 32-bit hex words for the packet. After input, the process will repeat once more for the final 5 hex words. Enter each of the following 5 word groups at the appropriate prompts:
   a. **0x40004001 0x910ec0a8 0x1463c0a8 0x14140800 0xf6fbf802**
   b. **0x09000001 0x00000000 0x00000000 0x00000000 0x00000000**

4.4.19 HNR Transmit

1. Enter the command: **8430-tool eth0 -H send**
2. On the **tcpdump** screen, verify that the frame was transmitted to the Linux host from the ARM target.

4.4.20 Jumbo Frames

Note that the Linux 2.6.13 kernel only supports 2048-byte jumbo packets.

1. On the Linux host enter the command: **ifconfig *ethN* mtu 3000**
2. On the Linux host enter the following commands:
   a. **tcpdump -lnei *ethN***
   b. **ping -s 2000 192.168.10.99**
3. Verify that no ICMP response packets are returned from the ARM target.
4. Enter the command: **8430-tool eth0 -J on**
5. Verify that all ICMP response packets are returned from the ARM target.
6. Enter the command: **8430-tool eth0 -J off**
7. Verify that ICMP response packets from the ARM target cease.

4.4.21 Headroom Watermark

1. Enter the command: **8430-tool eth0 -Q headroom get**
2. Enter the command: **8430-tool eth0 -Q headroom set 15**
3. Enter the command: **8430-tool eth0 -Q headroom get**
4. Verify the change.

To restore the default value for Linux, enter the command: **8430-tool eth0 -Q headroom set 4**

4.4.22 Add Padding

This procedure tests the addition of padding to small packets.

1. On the Linux host, enter: **ping -s 0 192.168.10.99**
2. On tcpdump, observe that the ping request from the Linux host has length 42 and the replay from the ARM target has length 60.
3. Enter the command: **8430-tool eth0 -F txpadding off**
4. On the Linux host, verify that the ping responses have stopped.
5. Enter the command: **8430-tool eth0 -F txpadding on**
6. On the Linux host, verify that the ping responses have resumed.

4.4.23 BIST

1. Enter the command: **8430-tool eth0 -B**
2. Verify successful tests in the BIST console log. The command generates five tests; all tests must report passed.

4.4.24 Classification Interrupts

This procedure verifies that Classification Interrupts cause an out of cycle RX routine call within the driver.

1. Start up TOE mode by entering: **8430-tool eth0 -T on**
2. Enter the command: **8430-tool eth0 -C new T 0x24 0xFE4AF040 0x0005009A**
3. On the Linux Host, enter: **ping 192.168.10.99**
4. Enter the command: **8430-tool eth0 -r debug 0x0000E008**
5. On the ARM target console log, verify detection of the Classification Interrupt by looking for the following message:

   net_interrupt: Classification Interrupt detected
6. Exit TOE mode after the test by entering: **8430-tool eth0 -T off**

4.4.25 CAM Reset

1. Attach an oscilloscope to the IRQ pin
2. Set the scope to 1 V and 10 µs
3. Set vertical cursors on
4. On the Linux Host, enter: **ping -f -s 0 192.168.10.99**
5. Measure the maximum duration of the IRQ assertion:

   Max Assert Interval: 20.8 µs; With step in de-assert at Minimum Interval: 16.6 µs
6. Start TOE mode by entering: **8430-tool eth0 -T on**
7. Measure the new duration of the IRQ assertion:

   TOE Assert Interval: 14.6 µs (decreased). Also, note the solid or nearly solid de-assertion without a step. It indicates a decrease in the IRQ code path length when TOE is active.
8. Reset the CAM: **8430-tool eth0 -C reset**
9. Observe the Oscilloscope. Verify that the IRQ assertion returns to the non-TOE duration, i.e. longer than in TOE mode and with a step in the de-assertion.

4.5 TOE Tests

The procedures in this section test the TCP offload engine frame transfer functions.

4.5.1 Fragmented ICMP – large, non-fragmented frames

1. Enter the command: **8430-tool eth0 -r debug 0x00000000**
2. Enter the command: **8430-tool eth0 -r debug 0x0000C000**
3. Enter the command: **8430-tool eth0 -T on**
4. Turn on jabber by entering: **8430-tool eth0 –J on**
5. On the Linux host, use an ethernet card whose driver supports MTUs > 1500, then enter the following commands:
   a. **tcpdump -lnei *ethN***
   b. **ifconfig *ethN* mtu 3000**
   c. **ping -s 2000 192.168.10.99**
6. Using **tcpdump**, verify that at least 2000 bytes are being sent by the Linux host and returned by the 78Q8430 in TOE mode.
7. On the ARM target, verify that the TOE XMIT is being invoked on classification index **0x78** by looking for the following messages**:**

   net_rx: RPSR Classification: 0x78

   net_rx: RPSR Class: 0x78,-> TOE XMIT

8. On the Linux Host, enter:
    a. **ifconfig *ethN* mtu 1500**
    b. **ping 192.168.10.99**
9. Verify that the 78Q8430 is responding normally. TOE mode will still be active.
    net rx: RPSR Classification: 0x78
    net_rx: RPSR Class: 0x78,-> TOE XMIT
10. Enter the command: **8430-tool eth0 -T off**
11. Verify that the 78Q8430 is responding normally.
    net_rx: RPSR Classification: 0x31

4.5.2 Fragmented ICMP – large, fragmented frames

1. Enter the command: **8430-tool eth0 -r debug 0x00000000**
2. Enter the command: **8430-tool eth0 -r debug 0x0000C000**
3. Enter the command: **8430-tool eth0 -T on**
4. On the Linux host...
    a. **tcpdump -lnei *ethN***
    b. **ifconfig *ethN* mtu 1500**
    c. **ping -s 2000 192.168.10.99**
5. Verify, on tcpdump that 1 frame of 1514 bytes and 1 frame of 562 bytes are being sent by the Linux host and returned by the 78Q8430. In this case “TOE XMIT” must not be seen in the console output.
6. On the ARM target, verify that the classification index is 0x0D by looking for the following message:
    net_rx: RPSR Classification: 0x0D
7. On the Linux host, enter: **ping 192.168.10.99**
8. Verify that the 78Q8430 is responding normally (TOE mode will still be active and processing the packets) by the presence of the following message:
    net_rx: RPSR Classification: 0x0D
9. On the Linux host, **ping 192.168.10.99** to verify that the 78Q8430 is responding normally. TOE mode will still be active and processing the packets.
    net_rx: RPSR Classification: 0x78
    net_rx: RPSR Class: 0x78 -> TOE XMIT
10. Enter the command: **8430-tool eth0 -T off**
    net_rx: RPSR Classification: 0x31
11. Verify that the 8430 is responding normally.

4.5.3 ICMP Checksum

1. Enter the command: **8430-tool eth0 -r debug 0x00000000**
2. Enter the command: **8430-tool eth0 -r debug 0x00010000**
3. Enter the command: **8430-tool eth0 -T on**
4. On the Linux Host, enter: **ping 192.168.10.99**
5. Verify that the 78Q8430 is responding normally. The following messages should be seen:
    toe_xmit: ICMP incoming Checksum: 0x46C2
    toe_xmit: ICMP outgoing Checksum: 0x4EC2

    The actual numeric values of the MSBs will vary from frame to frame. The incoming checksum will differ from the outgoing checksum by an addition of 0x0800 to the incoming checksum. The LSB must not change.
6. Enter the command: **8430-tool eth0 -T off**
7. Verify that the 8430 is responding normally.

4.5.4 IP Address Filter

1. Enter the command: **8430-tool eth0 -r debug 0x00000000**
2. Enter the command: **8430-tool eth0 -r debug 0x0003C000**
3. Enter the command: **8430-tool eth0 -T on**
4. On the Linux Host, define and ping a nonexistent IP address:
    a. **arp -s 192.168.10.98 00:01:02:03:04:05**
    b. **ping 192.168.10.98**
5. Verify that the 78Q8430 is dropping these packets; the following messages should be seen:
    toe_xmit: eth0: ICMP Addr: 0x6214A8C0, local addr: 0x6314A8C0.
    toe_xmit: eth0: ICMP Addr Mismatch: Dropping!
6. On the Linux Host, enter: **ping 192.168.10.99**
7. Verify that the 78Q8430 is responding normally.

5 Related Documentation

The following 78Q8430 documents are available from Teridian Semiconductor Corporation:

78Q8430 Preliminary Data Sheet
78Q8430 Layout Guidelines
78Q8430 Embest Evaluation Board User Manual
78Q8430 STEM Demo Board User Manual
78Q8430 Software Driver Development Guidelines
78Q8430 Linux Driver ARM Platform User Guide
78Q8430 ARM9(920T) Linux Driver Diagnostic Guide (this document)
78Q8430 Driver Manual for ST 5100/OS-20 with NexGen TCP/IP Stack

6 Contact Information

For more information about Teridian Semiconductor products or to check the availability of the 78Q8430, contact us at:

6440 Oak Canyon Road
Suite 100
Irvine, CA 92618-5201

Telephone: (714) 508-8800
FAX: (714) 508-8878
Email: lan.support@teridian.com

For a complete list of worldwide sales offices, go to http://www.teridian.com.

Appendix A – Acronyms

ARC	Address Resolution Controller
BIST	Built in Self Test
CAM	Content Addressable Memory
FTP	File Transfer Protocol
IOCTL	Input/Output Control
HNR	Host Not Responding
ICMP	Internet Control Message Protocol
IP	Internet Protocol
MAC	Media Access Control
PHY	Physical Layer Entity
PC	Personal Computer
RMON	Remote monitoring MIBS – belong to SNMP protocol family
SOC	System on Chip
TCP	Transport Control Protocol
TCP/IP	TCP over IP protocols which is the core protocol for internet communications
TOE	TCP Offload Engine
WOL	Wake-on-LAN

Revision History

Revision	Date	Description
1.0	6/2/2008	First Publication
1.1	10/17/2008	Corrected the document reference in Section 3. Removed the 8430-tool Command Summary section.