TERIDIAN SEMICONDUCTOR CORP.

Simplifying System Integration™

78Q8430 Linux Driver
ARM Platform User Guide

October 17, 2008
Rev. 1.0
UG_8430_008

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 4

2 Linux Driver 5

2.1 8430-tool Application 5

2.1.1 8430-tool Command Summary 6

2.1.2 8430-tool Command Descriptions 8

2.2 ethtool Command Syntax 16

2.3 mii-tool Command Syntax 16

3 8430-tool Debug Command Bit Mask Values 17

4 Related Documentation 20

5 Contact Information 20

Appendix A – Acronyms 21

Revision History 22

1 Introduction

The Teridian Semiconductor Corporation 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. 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 the Embest implementation of MIZI Research, 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. The acronyms used in this document are listed in the Acronyms section.

2 Linux Driver

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.

This section describes the user space command interfaces provided with the 78Q8430 Linux device driver. The user space command interfaces include:

- 8430-tool: for the chip features that cannot be addressed by the standard Linux Ethernet command interfaces.
- ethtool: for standard Linux Ethernet control commands.
- mii-tool: for standard Linux Ethernet control commands.

2.1 8430-tool Application

This section describes the 8430-tool usage. This tool supports many features of the 78Q8430 that are not covered by ethtool and mii-tool. The 8430-tool application provides support for the following 78Q8430 features:

- BIST execution and reporting.
- ARC definition and control.
- CAM definition and control.
- Firewall support.
- Frame modification and packet control.
- HNR support.
- WOL support for Magic Packet and OnNow.
- PAUSE packet support with user configurable parameters.
- RMON support.
- PHY/MII support (supplemented by ethtool and mii-tool).
- Jumbo packet support up to 8K on Linux.
- MAC reset.
- TCP Offload Engine (TOE) support for ICMP packets.
- Debug support with register access for MAC and PHY.

A summary of the commands and syntax is followed by a detailed description of each command.

2.1.1 8430-tool Command Summary

To display the list of 8430-tool commands and syntax, enter the following at the prompt:

~ $ **8430-tool -h**

```
8430-tool interface -option operand[s]
Options & Operands:
  B: BIST run all tests
  c: CAM query
       [types|map|filter|rule|irq|dump]
       types
       map
       irq
       filter _type_ [_filter_nmbr_]
       rule _nmbr_
       dump [_start_rule_ [_count_]]
  C: CAM control
       [irq|arc|new|rule|reset]
       arc [m|u] [ 2 | 1 thru 7 ] _MAC_addr_
       new _type_ _rule_ _rmr_ _rcr_
       rule _rule_ [on|off]
       irq [on|off]
       reset
  f: IP Firewall: [on|off]
  F: RX Frame Mods
       txpadding [on|off]
       strip [crc|pad] [on|off]
       crc [append|fix] [on|off]
       query [strip|crc]
  H: HNR: [set [0|1|2] _5_4-byte_words_|send]
  J:  Jabber: [on|off]
  L:  WOL
       onnow [on|off|set _data1_ _data2_ _data3_ _mask_]
       magic [on|off|set _MAC_addr_]
       pmectl [on|off]
       pmepwr[on|off]
  m: RMON counter query
  M: RMON counter control
       [on|off|clear]
       [set _reg_ offset [0|1] _value_]
  p: PHY control support
       link [100fd | 100hd | 10fd | 10hd]
       read _reg_
       write _reg_  _value_
       mdix [ [auto [on|off]] | on | off ]
       pwrsave [on|off] WARNING: For External Clock ONLY!
  P: PAUSE support
       [frame|macadr|time|threshold|local|send]
       frame [user|default]
       macadr _user_frame_MAC_addr_
       threshold n
       time n
       local [on|off]
       send
  Q: Thresholds:
       headroom  [get|set _value_]
       highwater [get|set _value_]
       irqdelay  [get|set _value_]
       txdivisor [get|set _value_]
  R: MAC Reset
  T: ICMP/TCP Offload Engine(TOE):
       on
       off
```

```
r: 8430 register access
     read  _reg_
     write _reg_ _32_bit_hex_value_
     debug _32-bit-debug_bit_mask_in_hex_
     status [rx|tx]
```

2.1.2 8430-tool Command Descriptions

This section provides detailed syntax and option descriptions for each 8430-tool command. The general command syntax is:

8430-tool eth*n* **-***option* [*operands*]

Keywords and options in bold must be entered as shown. The items in *italics* are replaced with user specific information. For **eth***n*, the *n* is replaced with the Linux PC ethernet port (0, 1, or 2) being used. For the procedures in this document, use the port number that has been assigned to the 78Q8430 board. *-option* is replaced with a command option and *[operands]* with operands for that command.

B: BIST run all tests

Syntax:

8430-tool eth*n* -B

Description:

This command option causes the driver to run all BIST tests.

Operands:

None.

c: CAM Query

Syntax:

8430-tool eth*n* **-c types**
map
irq
filter *type* [*filter_nmbr*]
rule *nmbr*
dump [*start_rule* [*count*]]

Description:

This command option causes the driver to display information about the CAM settings.

Operands:

types	Displays a list of CAM types defined by the driver.
map	Displays a map of CAM sections, types and rule counts.
irq	Displays the enablement state of the Classification Interrupt.
filter *type* [*filter_nmbr*]	Displays a list of CAM rules for a specified *type*. One of the types listed by the **types** operand, above. For **types** with multiple entries such as unicast ARC, the entry number within that type.
rule *nmbr*	Displays a single CAM rule *nmbr*.
dump [*start_rule* [*count*]]	Displays from one to all CAM rules. *start_rule* specifies the starting rule number to display. *count* specifies the number of rules to display starting with the *n*th rule.

C: CAM control

Syntax:

8430-tool eth*n* **-C arc** [**m|u**] [**2|7:1**] *MAC_addr*
new *type rule rmr rcr*
rule *rule_nmbr* [**on|off**]
irq [**on|off**]
reset

Description:

This command option causes the driver to modify CAM entries to define ARC entries made up of MAC addresses, to define or redefine CAM rules and to toggle on and off any CAM rule.

Operands:

arc [**m|u**] [**2|7:2**] *MAC_addr*	Keyword specifying the insertion of a MAC address into one of 6 unicast arc entries or into the single multicast arc entry. The first unicast arc entry is occupied by the MAC address of the local device. **m** indicates insertion into the multicast arc entry. **u** indicates insertion into one of the six unicast entries. **m** for multicast requires 2. **u** for unicast allows 7 down through 2. Note that unicast entry 1 is already occupied by the local MAC address. *MAC_addr* specifies the MAC address to be added to the ARC filter.
new *type rule rmr rcr*	Keyword specifying the overlay of a particular CAM rule. *type* is one of the types given by the **–c types** command described above. *rule* specifies the number of the rule to be modified or redefined. *rmr* specifies the CAM RMR register value. *rcr* specifies the CAM RCR register value.
rule *rule* [**on|off**]	Keyword to toggle a CAM rule on or off. *rule* specifies the number of the rule to be toggled on or off. [**on|off**] toggles the rule specified by **rule** *rule* on or off.
irq [**on|off**]	Toggles Classification Interrupt generation. **on** enables Classification Interrupt generation. **off** disables Classification Interrupt generation.
reset	Resets all CAM rules to the hardware default and then reinitializes the interface's MAC address as the first ARC entry. (The MAC address is normally taken from the EEPROM.)

f: IP Firewall

Syntax:

8430-tool eth*n* **-f** [**on|off**]

Description:

This command option causes the driver to activate or deactivate an IP firewall filter defined to block local host DOS attacks.

Operands:

[**on|off**]	Toggles the firewall filter on or off.

F: RX Frame Mods

Syntax:

8430-tool eth*n* **-F txpadding** [**on|off**]
strip [**crc|pad**] [**on|off**]
crc [**append|fix**] [**on|off**]
query [**strip|crc**]

Description:

This command option causes the driver to query and modify RX and TX frame handling.

Operands:

txpadding [**on|off**]	**on** enables transmission padding on small packets. **off** disables transmission padding on small packets.
strip [**crc|pad**] [**on|off**]	Keyword specifying the removal of CRC and/or padding from incoming RX packets. **crc** selects the removal of CRC or padding. **pad s**elects the removal of padding.
crc [**append|fix**] [**on|off**]	Keyword specifying appending or repairing the CRC on TX packets. Determines whether a TX CRC will be appended to outgoing frames (**append**) or whether an existing TX CRC will be recalculated (**fix**).
query [**strip|crc**]	Displays the state of the RX strip options or the TX CRC options. **strip** displays RX status**. crc** displays TX status.

H: Host Not Responding

Syntax:

8430-tool eth*n* **-H** [**set** [**0|1|2**] *5_4-byte_words***|send**]

Description:

This command option causes the driver to define an HNR frame and optionally send it manually as opposed to the default automatic delivery.

Operands:

set	Keyword specifying the total 60 byte HNR frame in three groups of five 32-bit words each.
[**0|1|2**]	Indicates which of the three groups of five 32-bit words is being specified.
5_4-byte_words	Five 32-bit words of the form 0xnnnnnnnn.
send	Manually send the HNR frame.

J: Jabber – Jumbo Frame Control

Syntax:

8430-tool eth*n* **-J [on|off**]

Description:

This command option causes the driver to activate or deactivate Jumbo packet support. The maximum jumbo packet size on Linux is 8K.

Operands:

[**on|off**]	Toggles Jumbo packet support on or off. The default is Jumbo support off.

L: WOL

Syntax:

8430-tool eth*n* **-L onnow** [**on|off|set** *data1 data2 data3 mask*]
magic [**on|off|set** *MAC_addr*]
pmectl [**on|off**]
pmepwr [**on|off**]

Description:

This command option causes the driver to configure and activate or deactivate the OnNow and Magic Packet WOL features.

Operands:

onnow [**on|off**]	Keyword to configure and activate OnNow WOL. Toggles OnNow on or off.
set	Establishes the OnNow 6 byte key from the three 32-bit input data words, *data1*, *data2* and *data3.* The key is selected from the 12 bytes of the three input data words based on the 12 least significant bits of the specified *mask* value. The command enters interactive mode for the input of data2, *data3* and *mask*. Each of the operands, 3 *data* and 1 *mask* **must** be specified as 32-bit hex numbers, of the form **0xnnnnnnnn**.
magic [**on|off**]	Keyword to configure and activate Magic Packet WOL. Toggles Magic Packet on or off.
set *MAC_addr*	Establishes the MAC address to be used for WOL recognition.
pmectl [**on|off**]	This keyword enables and disables the processor Power Management facility.
pmepwr [**on|off**]	This keyword enables and disables processor low power mode. The **on** value sets the processor into low power mode. The processor is then able to be awakened by one of the two 8430 WOL methods. Processor low power mode does not affect the power mode of the 8430.

m: RMON counter query

Syntax:

8430-tool eth*n* **–m**

Description:

This command option causes the driver to display all RMON counter values.

Operands:

None.

M: RMON counter control

Syntax:

8430-tool eth*n* **-M** [**on|off|clear**] [**set** *reg* **offset** [**0|1**] *value*]

Description:

This command option operates on the RMON counters.

Operands:

[**on|off|clear**]	Toggles the RMON counters on or off or clears all RMON counters.
set	Sets a given RMON register and its software 64-bit shadow register word to the specified value.
reg	RMON register number.
offset [**0|1**]	Each RMON register is 64 bits wide divided into two 32-bit words. 0 and 1 specify the high and low registers respectively.
value	The 32-bit hex value to be placed into the high or low register specified by **offset** above.

Both the high and low 32-bit words of the 64-bit register should be written. This requires two distinct command entries, one for **offset 0** and a second for **offset 1**.

p: PHY control support

Syntax:

8430-tool eth*n* **-p link** [**100fd | 100hd | 10fd | 10hd**]
read *reg*
write *reg value*
mdix [[**auto** [**on|off**]] **| on|off**]
pwrsave [**on|off**]

PWRSAVE **requires** external clocking. **Do not attempt** PWRSAVE if the 78Q8430 is configured for internal clocking.

Description:

This command option causes the driver to display or access the PHY registers or turn mdix, auto-negotiation or the power save state on or off.

Operands:

link [**100fd | 100hd | 10fd | 10hd**] Set the speed and duplex status of the link. Use of this command assumes that auto-negotiation is off. The available values are:

- **100fd** 100 Mbps, Full Duplex
- **100hd** 100 Mbps, Half Duplex
- **10fd** 10 Mbps, Full Duplex
- **10hd** 10 Mbps, Half Duplex

read	Keyword to display the contents of a PHY register.
reg	Specifies the PHY register to be read.
write	Keyword to modify the contents of a PHY register.
reg	Specifies the PHY register to be written.
value	The hex value to write to the register.
mdix	Modifies the negotiation state for RX/TX wire crossover.
auto [**on|off**]	Toggles auto-negotiation on or off. Auto-negotiation must be off to configure it manually.

[**on|off**] Toggles manual crossover state on or off.

pwrsave [**on|off**] Toggles PHY low power mode on or off. PWRSAVE requires external clocking. Do not attempt PWRSAVE if the 78Q8430 is configured for internal clocking.

P: PAUSE support

Syntax:

8430-tool eth*n* **-P frame** [**user|default**]
macadr *user_frame_MAC_addr*
threshold *n*
time *n*
local [**on|off**]
send

Description:

This command option causes the driver to configure and manage the PAUSE packet and its details.

Operands:

frame [user|default]	Keyword to specify default frame or user configuration. **user** selects user configuration. **default** selects the default frame.
macadr *user_frame_MAC_addr*	Keyword to set user frame MAC address. *MAC_addr* is a distinct MAC address to replace the standard PAUSE frame MAC address.
threshold *n*	Sets the RX Queue depth, *n*, beyond which a PAUSE frame will be sent automatically.
time *n*	Sets PAUSE duration, *n*, in 512-bit time units.
local [**on|off**]	Toggles the local PAUSE operation on or off.
send	Forces a PAUSE frame to be sent immediately. This option requires that the **frame** option be specified first.

Q: Thresholds

Syntax:

8430-tool eth*n* **-Q headroom** [**get|set** *value*]
highwater [**get|set** *value*]
irqdelay [**get|set** *value*]
txdivisor [**get|set** *value*]

Description:

This command option causes the driver to display or set the headroom, watermark interrupt and RX interrupt delay threshold values.

Operands:

headroom	Keyword to display or specify how many queue buffers to hold as overflow protection.
highwater	Keyword to display or specify how many queue buffers are to be consumed before a high watermark interrupt is generated.
irqdelay	Keyword to display or specify the RX interrupt delay.
txdivisor	Keyword to display or specify the TX completion frequency. The *value* is given by the equation: *value* = (2^n -1), for n, from 0 to 6, inclusive. This gives a minimum of 0 and a maximum of 63 (0x3F). TX completions are serviced on the completion of this number of transmissions.
[**get|set** *value*]	**get** displays the current value for the keyword it follows. **set** *value* specifies a new *value* for the keyword it follows.

R: MAC Reset

Syntax:

8430-tool eth*n* **-R**

Description:

This command option causes the driver to reset the MAC, which in turn results in a reset of the RX/TX Queues.

Operands:

None.

T: TCP Offload Engine (TOE)

Syntax:

8430-tool eth*n* **-T on|off**

Description:

This command option causes the driver to toggle TCP Offload Engine support for ICMP packets on and off.

Operands:

on|off	Toggles TOE offloading of ICMP packet processing on or off.

r: 8430 register access

Syntax:

8430-tool eth*n* **–r read** *reg*
write *reg 32_bit_hex_value*
debug *32-bit-debug_bit_mask_in_hex*
status [**rx|tx**]

Description:

This command option causes the driver to read and write on-chip registers and enable device driver debugging messages which are piped to the system console.

Operands:

read *reg*	Keyword to display the contents of a MAC register. *reg* is the register to be read.
write *reg 32_bit_hex_value*	Keyword to modify the contents of a MAC register. *reg* is the register to be written. *value* is the value to be written.
debug	Configures driver diagnostic message output based on the 32-bit hex bit mask, *32-bit-debug_bit_mask_in_hex*. Refer to Section 3, 8430-tool Debug Command Bit Mask Values, for a listing of the debug message masks.
status [**rx|tx]**	Display internal status information. **rx** shows receive information. **tx** shows transmit information.

2.2 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 ]
```

2.3 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 power on 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
```

3 8430-tool Debug Command Bit Mask Values

Operation of the 8430-tool Debug Command requires that the driver be compiled with the global debug option found in 78q8430-debug.h. This is the default state built into the executable. This option enables a set of prerequisite, corresponding options that are mapped to the 8430-tool Debug Command bit mask values specified below.

The command syntax for enabling the debug messages on the console is:

8430-tool eth*n* **-r debug** *0xXXXXXXXX*

The debug message bit masks are defined below. The 32-bit hex value above each group of debug messages is the mask to enable all of the messages in the group below it.

```
     unsigned int       if_debug;               // debug bit map
//                                      0x80000000
#define DEBUG_RMON                  ( 1 << 31)

//                                      0xE0000000
#define DEBUG_TX_INDEX               ( 1 << 30)
#define DEBUG_TX_RESEND              ( 1 << 29)
#define DEBUG_TX_SKB_RLSE            ( 1 << 28)

//                                      0x0F000000
#define DEBUG_RESEND_PKT             ( 1 << 27)
#define DEBUG_RESEND_PKT_ALIGNED     ( 1 << 26)
#define DEBUG_RESEND_PKT_TX_CNTS     ( 1 << 25)
#define DEBUG_RESEND_PKT_ALIGNED_XMIT  ( 1 << 24)

//                                      0x00800000
#define DEBUG_CAM            ( 1 << 23)

//                                      0x007E0000
#define DEBUG_SEND_PKT_ENT1_NDX      ( 1 << 22)
#define DEBUG_SEND_PKT_ENT2          ( 1 << 21)
#define DEBUG_SEND_PKT_ALIGNED       ( 1 << 20)

#define DEBUG_SEND_PKT_TX_CNTS       ( 1 << 19)
#define DEBUG_SEND_PKT_ALIGNED_XMIT  ( 1 << 18)
#define DEBUG_SEND_PKT_XMIT_DATA     ( 1 << 17)

//                                      0x00010000
#define DEBUG_TOE_XMIT_IPADDR        ( 1 << 16)   // TOE

//                                      0x0000E000
#define DEBUG_RX_CLASS               ( 1 << 15)   // RX classification results
#define DEBUG_RX_TOE                 ( 1 << 14)   // TOE classification hits
#define DEBUG_RX_DROP                ( 1 << 13)   // Packet drop w status & MTU

//                                      0x00001000
#define DEBUG_IOCTL             ( 1 << 12)   // RX loop top

//                                      0x00000FC0
#define DEBUG_RX_PKT_LEN             ( 1 << 11)   // Packet length
#define DEBUG_RX_RPSR                ( 1 << 10)   // RX RPSR (RX Pkt Status Word)
#define DEBUG_RX_SKB                 ( 1 <<  9)   // RX sk_buff (skb)
#define DEBUG_RX_SKB_IO              ( 1 <<  8)   // RX Slave DMA IO to skb
#define DEBUG_RX_SKB_DATA            ( 1 <<  7)   // RX skb data dump
#define DEBUG_RX_PROTO               ( 1 <<  6)   // RX protocol

//                                      0x00000020
```

```
#define DEBUG_PHY                       ( 1 <<  5)   // PHY init+ISR in -phy.c

//                                     0x0000001F
#define DEBUG_RX_INTERRUPT_WAKE         ( 1 <<  4)   // WOL Wake interrupt

#define DEBUG_RX_INTERRUPT_CLASS        ( 1 <<  3)   // Classification interrupt
#define DEBUG_RX_INTERRUPT_PHY          ( 1 <<  2)   // PHY      interrupt
#define DEBUG_RX_INTERRUPT_OFLOW        ( 1 <<  1)   // Overflow interrupt
#define DEBUG_RX_INTERRUPT_STATE        ( 1 <<  0)   // Interrupt cause register
```

The prerequisite #defines that enable the 8430-tool debug command are found in 78Q8430-debug.h.

Table 1 shows the relationship between the 78Q8430-debug.h #defines and the corresponding 8430-tool debug command debug flags, listed above.

Several 8430-tool debug command bit mask values gate multiple 78Q8430-debug.h #defines. The bit mask values in the table below, are shown with all the 78Q8430-debug.h #defines that are enabled by default. As an example, `DEBUG_PHY, 0x00000020,` controls output of console messages associated with 78Q8430-debug.h #defines.

Some messages have their #defines commented out, resulting in the corresponding message text and the corresponding 8430-tool debug flag test code, eliminated from the device driver object code, resulting in a smaller memory footprint.

More than one 8430-tool debug mask value may be controlled by a single 78Q8430-debug.h #define.

Also, a single 78Q8430-debug.h #define may generate two or more 8430-tool debug mask values.

Table 1: debug.h #defines and Debug Flags

Bit Mask Name	Header Debug Definition
DEBUG_RMON	#define DEBUG_NET_RMON
DEBUG_TX_INDEX	#define DEBUG_NET_TX_INDEX
DEBUG_TX_RESEND	#define DEBUG_NET_TX_RESEND #define DEBUG_NET_TX_RESEND_ERR
DEBUG_TX_SKB_RLSE	#define DEBUG_NET_TX_SKB_RLSE #define DEBUG_NET_TX_SKB_RLSE_ERR
DEBUG_RESEND_PKT	#define DEBUG_NET_RESEND_PACKET
DEBUG_RESEND_PKT_ALIGNED	#define DEBUG_NET_RESEND_PACKET_ALIGNED
DEBUG_RESEND_PKT_TX_CNTS	#define DEBUG_NET_RESEND_PACKET_TX_CNTS
DEBUG_RESEND_PKT_ALIGNED_XMIT	#define DEBUG_NET_RESEND_PACKET_ALIGNED_XMIT
DEBUG_CAM	#define DEBUG_NET_CAM_SET
DEBUG_SEND_PKT_ENT1_NDX	#define DEBUG_NET_SEND_PACKET_ENT1_NDX
DEBUG_SEND_PKT_ENT2	#define DEBUG_NET_SEND_PACKET_ENT2
DEBUG_SEND_PKT_ALIGNED	#define DEBUG_NET_SEND_PACKET_ALIGNED
DEBUG_SEND_PKT_TX_CNTS	#define DEBUG_NET_SEND_PACKET_ALIGNED

Bit Mask Name	Header Debug Definition
	#define DEBUG_NET_SEND_PACKET_TX_CNTS
DEBUG_SEND_PKT_ALIGNED_XMIT	#define DEBUG_NET_SEND_PACKET_ALIGNED_XMIT_DATA
DEBUG_SEND_PKT_XMIT_DATA	#define DEBUG_NET_SEND_PACKET_ALIGNED_XMIT_DATA
DEBUG_TOE_XMIT_IPADDR	#define DEBUG_NET_TOE_XMIT_IPADDR
DEBUG_RX_CLASS	#define DEBUG_NET_RX_CLASS
DEBUG_RX_TOE	#define DEBUG_NET_RX_TOE
DEBUG_RX_DROP	#define DEBUG_NET_RX_DROP
DEBUG_IOCTL	#define DEBUG_NET_IOCTL_ENTRY #define DEBUG_NET_IOCTL_PHY #define DEBUG_NET_IOCTL_MACRESET #define DEBUG_NET_IOCTL_REGS #define DEBUG_NET_IOCTL_THRESHOLDS #define DEBUG_NET_IOCTL_PAUSE #define DEBUG_NET_IOCTL_CAM #define DEBUG_NET_IOCTL_RMON #define DEBUG_NET_IOCTL_TOE #define DEBUG_NET_IOCTL_PRMSC
DEBUG_RX_PKT_LEN	#define DEBUG_NET_RX_PKT_LEN
DEBUG_RX_RPSR	#define DEBUG_NET_RX_RPSR
DEBUG_RX_SKB	#define DEBUG_NET_RX_SKB
DEBUG_RX_SKB_IO	#define DEBUG_NET_RX_SKB
DEBUG_RX_SKB_DATA	#define DEBUG_NET_RX_SKB
DEBUG_RX_PROTO	#define DEBUG_NET_RX_PROTO
DEBUG_PHY	#define DEBUG_NET_PHY_INIT_IRQ_SETUP #define DEBUG_NET_PHY_INIT_STATUS #define DEBUG_NET_PHY_INIT_IRQ_STATUS #define DEBUG_NET_PHY_INIT_TESTCTL #define DEBUG_NET_PHY #define DEBUG_NET_PHY_ISR #define DEBUG_NET_PHY_IO
DEBUG_RX_INTERRUPT_WAKE	#define DEBUG_NET_INTERRUPT_WAKE
DEBUG_RX_INTERRUPT_CLASS	#define DEBUG_NET_INTERRUPT_CLASS
DEBUG_RX_INTERRUPT_PHY	#define DEBUG_NET_INTERRUPT_PHY
DEBUG_RX_INTERRUPT_OFLOW	#define DEBUG_NET_INTERRUPT_OFLOW
DEBUG_RX_INTERRUPT_STATE	#define DEBUG_NET_INTERRUPT_STATE (Very high message volume)

4 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 Driver Manual for ST 5100/OS-20 with NexGen TCP/IP Stack
78Q8430 ARM9(920T) Linux Driver Diagnostic Guide
78Q8430 Linux Driver ARM Platform User Guide (this document)

5 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
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	10/17/2008	First publication.