Gigabit Ethernet/PCI Network Interface Card

Host/NIC Software Interface Definition

Revision 12.3.11

P/N 020001

Alteon Networks, Inc.
50 Great Oaks Blvd.
San Jose, CA 95119

1-408-360-5500

June 1999

Restricted Rights Legend

Revision History

Rev 01 6/3/96 Initial Alteon Networks version.

Rev 02 6/5/96 Added statistics, completely described commands and events, fixed many figures.

Rev 03 6/10/96 Major re-write, added flow diagrams, added CONFIDENTIAL markings, lots of changed based on feedback from Wayne and Ted, removed EEPROM layout.

Rev 04 6/19/96 Minor re-organization, removed FRAME_SENT and FRAME_RECEIVED events.

Rev 05 7/5/96 Updated data structures, statistics, removed START_FIRMWARE command, revised bibliography

Rev 06 9/26/96 Converted to framemaker, updated statistics and shared data structures, editorial changes.

Rev 07 11/8/96 Added SET_HOST_STATE command, rewrote initialization section, cleanup figures, update and expand command and event descriptions, Remove GET_STATS command, update for address changes related to RR->TIGON ASIC change, general editorial changes.

Rev 08 1/9/97 Defined all undefined values.

Rev 09 6/1/97 Changed format to comply with Alteon documentation standards, removed GET_RCB_FLAGS and SET_RCB_FLAGS commands, updated flow diagrams to current design, added section on performance tuning, general editorial changes.

Rev 10 9/1/97 Update document to Revision 1.1 specifications

Rev 10.1 9/17/97 Fix documentation bugs, particularly field definition errors in several of the data structure. Add new section on Serial EEPROM and new fields in the buffer descriptor.

Rev 10.2 10/21/97 Add documentation for new features, correct English and descriptions

Rev 11.0 11/19/97 Update document to Revision 2.1 specifications

Rev 11.1 12/15/97 Add documentation for checksum offload support

Rev 11.2 1/9/98 Documentation corrections and clarifications

Rev 11.3 2/123/98 Documentation corrections and clarifications. Tigon 2 checksum offload support.

Rev 12.0 4/15/98 Reorganized the shared memory areas. New VLAN assist functionality. Update to clock tick handling. Internal loopback available. Special monitoring capability.

Rev 12.0.1 - 12.0.5 5/19/98 - 6/23/98 Minor clarifications

Rev 12.0.6 6/26/98 Add DMA Errata for Tigon 2

Rev 12.1 - 7/29/98 Add BD Coal Now bit, RCB Coal update only bit.

Rev 12.1.1 - 8/4/98 Add SBUS bit to mode_status word

Rev 12.2 - 8/19/98 Add Host based Send Ring

Rev 12.2.1 - 8/24/98 Add mailbox receive ring replenish, extended multicast commands, new rupt control bits, eliminate receive return consumer index.

Rev 12.3 - 9/23/98 Add mini receive ring, extended receive buffer descriptors, transmit buffer ratio, and 802.3 checksum offload support documentation

Rev 12.3.3 - 1/15/99 Allow sizing of Receive Return Ring

Rev 12.3.9 - Better numbering scheme and fixed description of tcp_udp_cksum field in receive buffer descriptor

Rev 12.3.11 - Indicate Min DMA field must be zero due to hardware bug.

Table of Figures

Table of Tables

1 Introduction

The Alteon Networks Gigabit Ethernet/PCI Network Interface Card (NIC) provides a full or half duplex Gigabit Ethernet interface compliant with the proposed IEEE 802.3z standard. The NIC is implemented for use in systems that support either the 32 or 64 bit wide PCI Local Bus operating at 33 MHz. The Tigon 2 equipped version of the NIC adds support for 66 MHz buses. This manual also includes documentation for a 32-bit or 64-bit SBus card using the same Tigon technology with a special bridge chip.

The NIC acts as a bus master to DMA data between the Host and the NIC. Host access to the NIC is through a shared memory segment and control structures in Host memory. The NIC has either 1 MB or 512 KB of memory depending on the hardware configuration.

The NIC performs all the necessary Gigabit Ethernet MAC and PHY layer processing. The Host software provides frames to be sent and accepts received frames. Data is managed using ring buffers. Interrupts are generated based on conditions in the Event, Send or Receive Rings and not necessarily on a per frame basis.

1.1 TIGON

The Alteon Networks Gigabit Ethernet NIC is implemented by utilizing a proprietary custom Application Specific Integrated Circuit (ASIC). This ASIC, known as the Tigon ASIC or Tigon (for short), incorporates dual DMA channels, a Gigabit Ethernet MAC, the PCI Interface, and a RISC based processor. The Tigon 2 ASIC adds a second RISC processor, higher memory throughput, additional hardware offload features, and higher clock rates. Throughout this document the original Tigon chip will be referred to as Tigon or as Version 4 Tigon. The newer Tigon 2 chip will be referred to as Tigon 2 or Version 5 or Version 6 Tigon.

Version 4 (original Tigon ASIC) and Version 5 or 6 (Tigon 2 ASIC) usage are defined in this document. Different firmware for each version of the ASIC is provided in the Open Driver Kit. You can write your driver to support either ASIC or both ASICs. To support only the original version 4 ASIC please ensure that TIGON_REV is defined to be 1 either through an include file or through your C compiler and your Makefile. To support only Version 5 and 6 of the ASIC please ensure that TIGON_REV is defined to be 2. To support all of Version 4, Version 5, and Version 6 please ensure that TIGON_REV is defined to be 3.

1.2 Feature Set

The Alteon NIC and firmware allow a large number of host assistance features. One of these is checksum offload. The NIC can do complete checksum offload for both transmitted and received frames. This checksum offload feature allows full checksum calculation (including pseudo-headers) for IP, TCP, and UDP packets. On the transmit side the host can even set up the buffers in such a way that fragmented frames can have their UDP or TCP checksums calculated over the entire datagram. Detailed discussion of the checksum offload features can be found in section 2.6 of this document.

Other features designed to improve total throughput are available and are built into the API. These features include event coalescing and interrupt avoidance.

1.3 Foundation Data Structures

1.3.1 Host Addresses

The Host/NIC interface views all Host addresses as 64 bit objects, even on 32 bit hosts. On 32 bit Hosts, the upper 32 bits must be zero. These addresses are in big endian byte order.

1.3.2 NIC Addresses

All of the NIC address are 32 bits long and can only be addressed via 32 bit accesses with the exception of PCI Configuration Space and PCI Configuration Registers which can be addressed as 1, 2, 3, or 4 bytes.

NOTE: *Alteon Networks personnel have discovered that some PC debuggers while providing a 32 bit write command, actually perform four one-byte writes. This WILL NOT work.*

1.3.3 MAC Addresses

The MAC addresses are 48 bit objects which are padded to 64 bits as shown in Figure 1. [3].

31	23	15	7 0
0		octet 0	octet 1
octet 2	octet 3	octet 4	octet 5

Figure 1. MAC Address Format

1.3.4 Endianness

The NIC is a big endian machine. The DMA engine can be configured to perform byte swaps while transferring data, control information or both. All data DMAed to and from the NIC should be in Host byte order on the Host; it is byte swapped to the proper endian during DMA. All data in the shared memory space must be big endian since the NIC is big endian. The endian configuration of the control and data DMA transfers is set by the Host in the Operating Mode and Status register (See “Operating Mode”, section 2.1.1.4.4). In addition, all data that is placed into the NIC through PIOs needs to be in big endian format. This is controlled through the use of the Miscellaneous Host Control register (See “Tigon Miscellaneous Host Control Register”, section 2.1.1.2.1).

There is also the concept of “word endianness” in the NIC. Data is either dealt with as a 32 bit quantity or as a byte stream. In either case, since the memory subsystem of the NIC is a 64 bit memory subsystem, the host must ensure that the words are reversed. This is done by setting the proper word swapping bits in the Misceallaneous Host Control register (See “Tigon Miscellaneous Host Control Register”, section 2.1.1.2.1) and the Operating Mode and Status register (See “Operating Mode”, section 2.1.1.4.4).

The following is a brief discussion to help you figure out how to set these bytes on your system.

Rule 1: Since our basic unit of interaction is a 32 bit quantity the bits show up the same whether they are big-endian or little-endian.

Rule 2: PCI is defined as a little-endian bus.

Given these rules, lets look at what typically happens on little-endian hosts. Since the PCI bus is defined as little-endian and the memory system is defined as little-endian most (all?) little-endian systems pass the bytes through to the bus in the same order that they appear in memory. So for 32 bit quantities (as everything in the data structures is treated) there is no need to byte swap (the Misc Host Control register is set to not swap and the TG_CFG_MODE_SWAP_BD bit is not set).

For network data (which is defined as a byte stream) the data is presented in little-endian format (byte swapped) so the data must be byte swapped (TG_CFG_MODE_SWAP_DATA).

Now, let's look at "most" big-endian hosts. Since the PCI bus is defined as little-endian, and the memory system is defined as big-endian, the host is often built to "pre-swap" all the data that travels across the bus so it shows up in the "right order". But this swapping is really "wrong" for 32 bit quantities (we noted in rule 1 that these are always the same). So in this case we have to set byte swapping on for all data structures (the byte swapping bit in the Misc Host Control register and the TG_CFG_MODE_SWAP_BD bit). Data is still byte oriented, but it is in big-endian format. Since these hosts pre-swap the data we must also byte swap data in order to get it back into the right order (hence the TG_CFG_MODE_SWAP_DATA bit).

One more endian issue has to do with word swapping. This can effect both addresses in the shared memory region and those in the host. For little endian systems that have no way of doing a 64 bit write to the shared memory put support 64 bit hardware address (e.g. some Intel x86 systems) the high order 32 bits will have to be swapped with the low order 32 bits. This is because the NIC wants all 64 bits in big endian format and the host has them in little endian format. If a 64 bit transfer were done then the words would be swapped by the hardware, but if no 64 bit instruction can be used the host code must explicitly swap them. For little endian systems with 64 bit addresses the two words of addresses in host memory will have to be swapped to create a big endian look in the host.

1.4 LED Meanings

There are two LEDs on Gigabit NICs. A “Link” LED and a “Data” LED. In general, the link LED indicates that the link is up and the data LED flashes when there is data movement on the NIC. The NIC enabled means that a TG_CMD_HOST_STATE UP command has been issued to the NIC (See “TG_CMD_HOST_STATE”, section 3.2.2).

Table 1. LED Usage

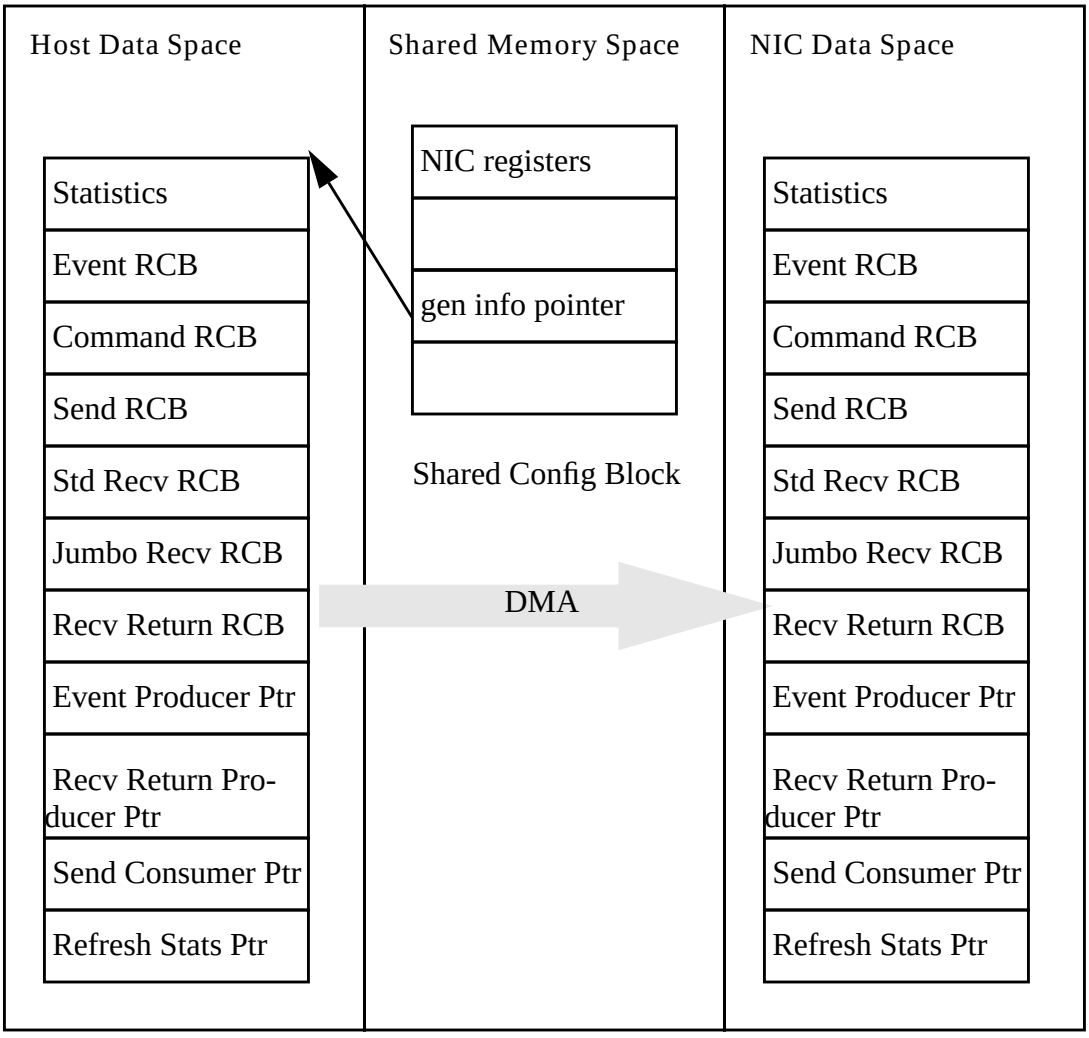
Link LED solid	Data LED with Solid Link LED	Data LED by itself	Slow Flashing Link LED
Link is active, NIC is enabled	Data activity on the NIC	Fault condition, firmware is halted	NIC is disabled, but firmware is loaded and running

2 Architecture

2.1 Memory Model

The NIC manages data structures in Host memory using DMA and the following two control blocks:

- the Shared Configuration Block, and
- the General Information Block.



2.1.1 Shared Configuration Block

This is a 16K byte shared memory region of Host mapped memory as shown in Table 2.

NOTE: *This region contains registers and "holes". There are addresses that cannot be read nor written.*

NOTE: *Bits and fields that are meaningful only for Tigon 2 are indicated with the words (Tigon 2 only) and are lightly shaded if described in a Table.*

All registers in this block can be shared between the Host and the NIC, however some are used operationally by the Host while others are used operationally by the NIC. All registers are 32 bits long.

Table 2. Shared Configuration Block

Shared Configuration Block	Address
PCI Configuration Region	0x0000 - 0x003F
Tigon Configuration Region	0x0040 - 0x04FF
Mailboxes	0x0500 - 0x05FF
General Communications Region	0x0600 - 0x07FF
Local Memory Window	0x0800 - 0x0FFF
Reserved	0x1000 - 0x3FFF

2.1.1.1 PCI Configuration Region

Please refer to the PCI Local Bus Specifications[1] for details. Vendor Subsystem IDs are supported only in Tigon 2 (Version 5 or 6) ASICs

2.1.1.1.1 PCI Configuration Region Setup for SBus Cards

Since the SBus card uses a bridge chip, the PCI bus configuration region will not be set up by the system's BIOS. The following registers must be set correctly. The SBus bridge chip only allows accesses in 4 byte quantities. All these registers should be set using the PCI Config region of the SBus address space (See "SBus Memory Model", section 2.2).

Table 3. PCI Configuration Registers for SBus Cards

PCI Configuration Registers	Address
Command/Status Register	0x04 - 0x07
Latency Timer, etc.	0x0C - 0x0F
Shared Memory Base	0x10 - 0x13

2.1.1.1.1.1 Command/Status Register

This register is a pair of registers that indicate the Status and the Command registers of the PCI Configuration space. This should be set to the value 0x02000006 in big endian format.

2.1.1.1.1.2 Latency Timer, etc. Register

This register is really four PCI configuration registers. It should be set to 0xFFFFFFFF.

2.1.1.1.1.3 Shared Memory Base Register

This register specifies the addresses that the SBus - PCI bus chip will generate when data is placed in the 16 KB shared memory region of the SBus address space. It also specifies to the PCI chip where it should respond to shared memory accesses. It is critical that this space not ever be used as a DMA address or the PCI chip will respond to its own DMA requests.

Table 4. Shared Memory Base Register

Bit	Name	Access	Description
0x000000FF	Reserved	W/O	These bits must be set to zero.
0x00000100	Chaining	W/O	If set the NIC will not perform SBus chaining.
0x00000600	Transfer Size	W/O	If ‘00’ do 16 word/dword transfers, if ‘01’ do 8 word/dword transfers, if ‘10’ do 4 word/dword transfers, if ‘11’ do 1 word/dword transfers.
0x00000800	Reset	W/O	If set the bridge chip is held in reset.
0x00003000	Reserved	W/O	This bits must be set to zero.
0xFFFFC000	Base Address	R/W	These bits specify the base address used by the bridge chip and the Tigon chip for shared memory accesses.

2.1.1.2 Tigon Configuration/Control Region

The Tigon Configuration/Control Region is composed of several 32 bit registers. See Table 5 for details.

.

Table 5. Tigon Configuration/Control Region

Tigon Configuration Region	Address
Tigon Miscellaneous Host Control	0x040 - 0x043
Tigon Miscellaneous Local Control	0x044 - 0x047
Tigon Miscellaneous Configuration (Tigon 2 only)	0x050 - 0x053
Tigon PCI State	0x05C - 0x05F
Tigon Window Base Address	0x068 - 0x06B
Tigon Window Data	0x06C - 0x06F
Tigon DMA Assist State	0x11C - 0x11F
Tigon CPU State	0x140 - 0x143
Tigon CPU Program Counter	0x144 - 0x147
Tigon Internal SRAM Address	0x154 - 0x157
Tigon Internal SRAM Data	0x158 - 0x15B
Tigon General Purpose Registers	0x180 - 0x1FF
Tigon MAC RX State	0x220 - 0x223
Tigon CPU Control (CPU B) (Tigon 2 only)	0x240 - 0x243
Tigon CPU Program Counter (CPU B) (Tigon 2 only)	0x244 - 0x247
Tigon Internal SRAM Address (CPU B) (Tigon 2 only)	0x254 - 0x257
Tigon Internal SRAM Data (CPU B) (Tigon 2 only)	0x258 - 0x25B
Tigon General Purpose Registers (CPU B) (Tigon 2 only)	0x280 - 0x2FF

2.1.1.2.1 Tigon Miscellaneous Host Control Register

The Miscellaneous Host Control register is used to control various functions within the Tigon normally controlled from the host interface.

All bits of the Miscellaneous Host Control register not specified in Table 6 must be set to 0.

Table 6. Tigon Miscellaneous Host Control Register

Bit	Name	Access	Description
0x00000001	Interrupt State	R/O	This bit reflects the state of the PCI IntA Output pin
0x00000002	Clear Interrupt	W/O	This bit clears the host PCI Interrupt IntA
0x00000008	Reset	R/W	Force a complete hardware reset, except PCI Configuration Registers (0x0 - 0x3F)
0x00000010	Enable Endian Byte Swap	R/W	Enable endian byte swapping when accessing Tigon through target interface
0x00000020	Enable Endian Word Swap	R/W	Enable endian word swapping when accessing Tigon through target interface.
0x00000040	Mask Interrupts	R/W	Mask interrupts while set. Pending interrupts will be generated when the bit is cleared. If the interrupt line is asserted it will be deasserted when this bit is set. (Tigon 2 Revision 6 only)
0xF0000000	Chip Version Mask	R/O	This is a 4-bit field which carries the Version of the Tigon ASIC. It can contain 4 (Tigon) or 5 or 6 (Tigon 2)

2.1.1.2.2 Tigon Miscellaneous Local Control Register

The Miscellaneous Local Control register is used to control various functions within the Tigon normally controlled from the internal processor. All bits are set to zero during reset.

Change only the Miscellaneous Local Control register bits specified in Table 7. All other bits must be left unchanged.

Table 7. Tigon Miscellaneous Local Control Register

Bit	Name	Access	Description
0x00000010	Enable EEPROM Write	R/W	Enables writing to EEPROM Region
0x00000200	SRAM Bank 256K	R/W	Each bank of SRAM is 256K instead of 1M bytes (Tigon 1 only)
0x00000300	SRAM Bank Size	R/W	A two bit field to indicate the size of the SRAM bank. 0 is for a single bank. 1 is for a bank size of 1M. 2 is for a bank size of 512KB. 3 is for a bank size of 256KB.PCI NICs have 512KB per bank. (Tigon 2 only)
0x00004000	Local Address21	R/W	Used for bank switching flash memory by directly controlling local address bit 21
0x00008000	Local Address22	R/W	Used for bank switching flash memory by directly controlling local address bit 22
0x00080000	SBus Write Error	R/O	Indicates there was an SBus Write Error
0x00100000	Serial EEPROM clock Output	R/W	Directly controls the clock output pin
0x00200000	Serial EEPROM Data Output Enable	R/W	When Asserted, Tigon drives the value of Serial EEPROM data output.

Table 7. Tigon Miscellaneous Local Control Register

Bit	Name	Access	Description
0x00400000	Serial EEPROM Data Output	R/W	Value of data to drive out when output enabled
0x00800000	Serial EEPROM Data Input	R/O	Input from bi-directional serial EEPROM data pin.

NOTE: *Writes to the Tigon Miscellaneous Local Control Register should only be performed when the internal processor is halted.*

2.1.1.2.3 Tigon Miscellaneous ConfigurationRegister

The Miscellaneous Configuration register is used to control various functions within the Tigon normally controlled from the internal processor at configuration time. All bits are set to zero during reset. This register is only available on the Tigon 2 ASIC.

Change only the Miscellaneous Configuration register bits specified in Table 7. All other bits must be left unchanged.

Table 8. Tigon Miscellaneous Configuration Register

Bit	Name	Access	Description
0x00100000	Set Synchronous SRAM timing	R/W	Must be set for boards that contain synchronous SRAM (Tigon 2 only)

NOTE: *Writes to the Tigon Miscellaneous Configuration Register should only be performed when the internal processor is halted.*

2.1.1.2.4 Tigon PCI State Register

The PCI State register is used to control various functions within the Tigon associated with PCI Interface. All bits are set to zero during reset.

All of the PCI State register bits not specified in Table 9 must be set to 0.

Table 9. Tigon PCI State Register

Bit(s)	Name	Access	Description
0x00000001	Force Reset	W/O	Will force an immediate reset of the PCI Interface. All state information within the PCI Configuration registers will be lost. This bit is self clearing.
0x00000002	Provide Length	R/W	Use non-standard PCI interface which provides transfer length indication. On PCI cards there is never any reason to set this bit.
0x0000001C	Read Max DMA	R/W	This is a 3-bit field. Encoded bits which force termination of PCI read operations at any of the following boundaries: Disable (000), 4 (001), 16 (010), 32 (011) , 64 (100), 128 (101), 256 (110) , 1K (111). For SBus cards this should be set to 100.

Table 9. Tigon PCI State Register

Bit(s)	Name	Access	Description
0x000000E0	Write Max DMA	R/W	This is a 3-bit field. Encoded bits which force termination of PCI write operations at any of the following boundaries: Disable (000), 4 (001), 16 (010), 32 (011), 64 (100), 128 (101), 256 (110), 1K (111). For SBus cards this should be set to 100.
0x0000FF00	Minimum DMA	R/W	This 8 bit field contains the minimum number of PCI words each DMA channel is allowed to keep the PCI bus without allowing accesses by the other DMA channel. This guarantees a minimum PCI usage rather than the usual alternate per burst behavior. Due to hardware bugs this field MUST be set to zero.
0x00010000	FIFO Retry enable	R/W	Enable PCI retry response to PCI target accesses to FIFO when FIFO cannot complete operation. This bit is not used for normal operation.
0x00020000	Use Mem_Read_Multiple PCI Command	R/W	Use Memory Read Multiple command in place of Memory Read Line for DMA reads.
0x00040000	No Word Swap (Read) DMA	R/W	Disable word swapping for all read DMA Transactions. For Tigon 2 ASICs this controls both the read and the write DMA channels.
0x00080000	No Word Swap Write DMA	R/W	Disable word swapping for all write DMA Transactions. Only for the original Tigon ASIC.
0x00080000	66 MHz Bus indication	R/O	Set if the card has negotiated for a 66 MHz bus clock speed. Tigon 2 ASIC only.
0x00100000	32-bit PCI bus	R/O or R/W	Current Host is providing only a 32-bit PCI Bus. On Tigon 2 ASICs can be written to override systems where 64 bit operation has been negotiated incorrectly.
0x00800000	Enable all Byte Enables	R/W	This bit must be used with Tigon 2 ASIC SBus cards and must not be set on any other card. It causes all the PCI byte enables to be set on every WRITE DMA.
0x0F000000	PCI Read Command	R/W	This is a 4-bit field. The NIC uses this command for all PCI read transactions of <3 words or half the cache line size if set. This field should normally be set to 6 (Memory Read).
0xF0000000	PCI Write Command	R/W	This is a 4-bit field. The NIC uses this command for all PCI write transactions. This field should normally be set to 7 (Memory Write). See “PCI Command Memory Write and Invalidate”, section 4.6 for information on using the Memory Write and Invalidate.

2.1.1.2.5 Tigon Window Base Address Register

The Window Base Address register defines the local memory address (from the point of view of the NIC’s internal processor) which is the base for the 2 KByte window provided by the Tigon (See “Shared Configuration Block”, section Table 2. and See “Local Memory Window”, section 2.1.1.5.) This register may contain any valid local memory address, but the usage of the least

significant 11-bits varies depending on how the local memory is addressed. If the 2 KByte window is used, then the least significant 11-bits are ignored and are substituted with zeros. If Window Data register is referenced, then the entire Window Address register is used to indicate the local memory address of the operation.

NOTE: *The Window Base Address register cannot be used by the host to access registers internal to Tigon. It is only used to access memory internal to Tigon.*

The Window Base Address register (in conjunction with the local memory window) is used to download the firmware into the NIC (See “Firmware Download”, section 5.4). It is also used to address the Send Ring. (See “Send Ring”, section 2.4.2.1.)

2.1.1.2.6 Tigon Window Data Register

The Window Data register is normally used to access locations in the local memory when the actual 2k Byte local memory window provided by the Tigon is unavailable. Only 32-bit operations are supported on this register.

The Window Data register (in conjunction with Window Base Address Register) provides an indirect method to access the entire local memory address space.

2.1.1.2.7 Tigon DMA Assist State Register

The DMA Assist State register is used to control the DMA assist logic within the Tigon. The DMA assist logic allows the NIC to queue DMAs that will then be dequeued and executed by the hardware. Stopping the DMA Assist does not stop the current DMA it only prevents the next queued DMA from starting.

All of the DMA Assist State register bits not specified in Table 10 must be left unchanged.

Table 10. Tigon DMA Assist State Register

Bit	Name	Access	Description
0x00000001	Enable	R/W	Enable DMA Assist Logic
0x00000002	Pause	R/W	Stop DMA Assist Logic. Does not affect the DMA Transfers which are active.

2.1.1.2.8 Tigon CPU State Register

The CPU State register controls miscellaneous functions associated with the CPU in addition to indicating the state of the processor. Alteon Networks uses these bits a great deal to implement a firmware debugger. In general, a driver writer should not need to adjust these bits directly. On the Tigon 2 ASIC there are two CPUs (CPU A and CPU B). There is a CPU State Register for each CPU.

Table 11. Tigon CPU State Register

Bit	Name	Access	Description
0x00000001	Reset CPU	R/W	Self clearing bit which resets the internal CPU.
0x00000002	Single Step CPU	R/W	Advances the CPU’s Program Counter one cycle. If the halting condition (e.g. an invalid instruction at the location) still exists then the CPU will again halt, otherwise it will resume normal operation.

Table 11. Tigon CPU State Register

Bit	Name	Access	Description
0x00000010	ROM Fail	R/W	Asserted on reset and cleared by ROM code after successfully loading code from serial EEPROM or Flash.
0x00010000	Halt CPU	R/W	Set by the external Host to halt the internal CPU.

2.1.1.2.9 Tigon CPU Program Counter Register

The CPU Program Counter register points to the instruction which is being fetched by the internal processor. It normally changes every clock cycle during normal execution. This register can be written by the host only when the processor has been halted. On the Tigon 2 ASIC there are two CPUs (CPU A and CPU B). There is a CPU Program Counter for each CPU.

2.1.1.2.10 Tigon Internal SRAM Address Register

This register is analogous to the Tigon Window Base Register for the accessing the Internal SRAM. One difference is that the memory can only be accessed through the Tigon Internal SRAM Data Register. That is, there is no equivalent to the Local Memory Window. On the Tigon 2 ASIC there are two CPUs (CPU A and CPU B). There is a Tigon Internal SRAM Address Register for each CPU. The size of the Internal SRAM is 2 KB on the original Tigon, 16 KB on CPU A of Tigon 2, and 8 KB on CPU B of Tigon 2. The local memory address of the Internal SRAM begins at 0xC00000.

2.1.1.2.11 Tigon Internal SRAM Data Register

This register is analogous to the Tigon Window Data Register.

2.1.1.2.12 Tigon General Purpose Registers

You can access the internal registers of the Tigon's CPU via the register file. There are 32 registers. On the Tigon 2 ASIC there are two CPUs (CPU A and CPU B). There is a CPU Register File for each CPU.

2.1.1.2.13 Tigon MAC RX State Register

The MAC RX state register is used to control and monitor the Ethernet receive interface of the Tigon.

All of the MAC RX state register bits not specified in Table 12 must be left unchanged.

Table 12. Tigon MAC RX State Register

Bit	Name	Access	Description
0x00000004	Stop RX After Next Packet	R/W	Setting this bit disables the Receive MAC gracefully at next inter-packet gap.

2.1.1.3 Mailboxes

This is a 256 byte region which contains 32 sixty-four bit registers. These registers are called Mailbox Registers (or Mailboxes). When a value is stored in the least significant 32 bits of these registers, an event (known as Mailbox Event) is generated to the Tigon Internal Processor.

In the present version of the API, only the first six Mailboxes are used. Others are reserved for future enhancements.

Table 13. Mailbox Registers

Mailboxes	Address
"Host In Interrupt Handler" (Mailbox 0)	0x500 - 0x507
Command Producer Index (Mailbox 1)	0x508 - 0x50F
Send Producer Index (Mailbox 2)	0x510 - 0x517
Standard Receive Producer Index (Mailbox 3) (Tigon 2 only)	0x518 - 0x51F
Jumbo Receive Producer Index (Mailbox 4) (Tigon 2 only)	0x520 - 0x527
Mini Receive Producer Index (Mailbox 5) (Tigon 2 only)	0x528 - 0x52F
Reserved (Mailbox 6- 31)	0x530 - 0x5FF

2.1.1.3.1 Host In Interrupt Handler

The NIC firmware uses this register to determine if it should interrupt the Host. A value of 1 in this register means the Host is presently handling Events, and should not be interrupted. Every time the Host enters the Interrupt Handler, this register must be set to 1. Once the Host is done processing all the events, it must set this register to 0. This register must be initialized to 0.

NOTE: *Writing any value into Mailbox Register 0 (the Host In Interrupt Handler mailbox) has a side effect of clearing the PCI Interrupt.*

2.1.1.3.1.1 Command Producer Index

The Command Producer Index register (Mailbox 1), contains the index of the next command in the Command Ring that will be produced. Host software writes this register whenever the command ring is updated. This register must be initialized to 0.

2.1.1.3.1.2 Send Producer Index

The Send Producer Index register (Mailbox 2), contains the index of the next buffer descriptor that will be produced. Host software writes this register whenever the send ring is updated. This register must be initialized to 0.

2.1.1.4 General Communications Region

These registers are active when the firmware is loaded and running. The registers are used in communication between the host and the processor internal to the Tigon chip. The area is 512 bytes long. Data in this area must be accessed 32 bits at a time.

Table 14 shows the mapping of registers in this area.

Table 14. General Communications Registers

General Communications Registers	Address
MAC address	0x600 - 0x607
General Information Pointer	0x608 - 0x60F
Multicast MAC Address Transfer Buffer	0x610 - 0x617
Operating Mode and Status	0x618 - 0x61B
DMA Read Configuration	0x61C - 0x61F

Table 14. General Communications Registers

General Communications Registers	Address
DMA Write Configuration	0x620 - 0x623
Transmit Buffer Ratio	0x624 - 0x627
Event Consumer Index	0x628 - 0x62B
Command Consumer Index	0x62C - 0x62F
Tuning Parameters	0x630 - 0x64F
NIC Trace Pointer	0x650 - 0x653
NIC Trace Start	0x654 - 0x657
NIC Trace Length	0x658 - 0x65B
ifIndex	0x65C - 0x65F
ifMtu	0x660 - 0x663
Mask Interrupts	0x664 - 0x667
Gigabit Link State	0x668 - 0x66B
10/100 Link State	0x66C - 0x66F
Reserved(4)	0x670 - 0x67F
Receive Return Consumer Index	0x680 - 0x683
Reserved(31)	0x684-0x6ff
Command Ring	0x700 - 0x7FF

2.1.1.4.1 MAC Address

The MAC Address register contains the MAC address of the NIC. It is in MAC address format (see Figure 1). This register must be initialized by the Host prior to starting the firmware. It is usually the MAC address extracted by the Host from the NIC's EEPROM.

The MAC address placed here is only read at two times by the NIC firmware. It is read at initialization and it is read when a TG_CMD_SET_MAC_ADDR command is issued (See "TG_CMD_SET_MAC_ADDR", section 3.2.12.)

2.1.1.4.2 General Information Pointer

The General Information Pointer register contains the 64 bit Host address of the General Information page in Host memory. This register must be initialized by the Host prior to starting the firmware. Once the firmware has initialized, it does not reference the General Information pointer again.

2.1.1.4.3 Multicast MAC Address Transfer Buffer (OBSOLETE)

The Multicast MAC Address Transfer Buffer is where the host places the multicast MAC Address to be added or deleted from the NIC's multicast list, using the TG_CMD_ADD_MULTICAST_ADDR or the TG_CMD_DEL_MULTICAST_ADDR commands. It is in MAC address format (see figure 1).

The use of this buffer has been obsoleted by the new TG_CMD_EXT_ADD_MULTICAST_ADDR and the TG_CMD_EXT_DEL_MULTICAST_ADDR commands.

2.1.1.4.4 Operating Mode

The Operating Mode register sets various operational modes in the firmware.

Table 15. Operating Mode and Status

Bit	Name	Description
0x00000002	BYTE_SWAP_BD	Byte swap buffer descriptors and events when DMAing them to and from the Host.
0x00000004	WORD_SWAP_BD	IThis bit is not operational in the current release of firmware, but should be set to one in case it should ever be enabled in future revisions.
0x00000008	WARN	Enables warnings through events from the firmware. There are no operational differences when setting this bit currently. It is recommended that this bit be set for possible future enhancements to the firmware.
0x00000010	BYTE_SWAP_DATA	Byte swap data when DMAing it to and from the Host.
0x00000040	1_DMA_ACTIVE	Program the interface to activate only one DMA Channel at a time, i.e. either Read or Write at a time, not both. It is recommended that this bit not be set. On SBus drivers, this bit MUST be set to One.
0x00000100	SBUS	Must be set for SBus cards. Must not be set for non-SBus cards.
0x00000200	DONT_FRAG_JUMBOS	Don't split received Jumbo frames across multiple standard sized buffer descriptors if no jumbo sized buffer descriptors are available.
0x00000400	INCLUDE_CRC	Include the CRC on all packets passed to the host and in the length field of the buffer descriptor.
0x00000800	ALLOW_BAD_FRAMES	DMA bad frames and associated receive buffer descriptors for frames that have errors. The error is indicated in the receive buffer descriptor. (See See "Receive Buffer Descriptors", section 3.1.1.)
0x00001000	DONT_RUPT_EVENTS	Never generate interrupts when the event producer index is updated.
0x00002000	DONT_RUPT_SENDS	Never generate interrupts when the send consumer index is updated.
0x00004000	DONT_RUPT_RECVS	Never generate interrupts when the receive return producer index is updated.
0x40000000	FATAL	Enables fatal error indication through events. There are no operational differences when setting this bit currently. It is recommended that this bit be set for possible future enhancements to the firmware

2.1.1.4.5 DMA Read Configuration

The NIC supports both 32-bit and 64-bit interfaces. Some Hosts may want to use 32-bit interface while others may want 64-bit mode. The DMA Read Configuration register allows the Host to program the interface appropriately. Additionally, the this register offers flexibility to control when the DMA engine should make a request for PCI operation.

The DMA read configuration register must be set by the Host based upon the interface it wishes the NIC to use while reading data from the Host Memory. Please refer to Table 16 for the appropriate values.

Table 16. DMA Read Configuration Mode

Bit(s)	Name	Description
0x00000008	FORCE_32_BIT_PCI	If set, force PCI Operation to be performed as if on a 32-bit PCI Bus. By default the bus will “self detect.”
0x000001F0	Threshold	This is a five bit field and represents the number of empty words in the FIFO before the DMA channel requests a PCI operation. The Host must set a value between 1 - 16. It is recommended to use a value of 8. SBus Driver Note: Use threshold value of 8 bytes for SBus Drivers.

2.1.1.4.6 DMA Write Configuration

The NIC supports both 32-bit and 64-bit interfaces. Some Hosts may want to use 32-bit interface while others may want 64-bit mode. The DMA Read Configuration register allows the Host to program the interface appropriately. Additionally, the this register offers flexibility to control when the DMA engine should make a request for PCI operation.

The DMA Write Configuration register must be set by the Host based upon the interface it wishes the NIC to use while writing data into the Host Memory. Please refer to Table 17 for appropriate values.

Table 17. DMA Write Configuration Mode

Bit(s)	Name	Description
0x00000008	FORCE_32_BIT_PCI	If set, force PCI Operation to be performed as if on a 32-bit PCI Bus. By default the bus will “self detect.”
0x000001F0	Threshold	This is a five bit field and represents the number of empty words in the FIFO before the DMA channel requests a PCI operation. Once the last word to be written to the host has been placed into the FIFO, the host bus will be requested until the FIFO goes empty. The Host must set a value between 1 - 16. It is recommended to use a value of 8. SBus Driver Note: Use threshold value of 8 bytes for SBus Drivers.

2.1.1.4.7 Transmit Buffer Ratio

This register is used to indicate the ratio of the remaining memory in the NIC that should be devoted to Transmit Buffer vs. Receive Buffer. The bottom 7 bits are used to indicate the ratio in 1/64th increments. For example, setting this value to 16 will set the transmit buffer to 1/4 of the remaining buffer space. In no cases will the Transmit or Receive buffer be reduced below 68 KB. For a 1 MB NIC the approximate total space for data buffers is 800 KB. For a 512 KB NIC that number is 300 KB.

2.1.1.4.8 Event Consumer Index

The Event Consumer Index register contains the index of the next event in the Event Ring that will be consumed. The Host writes this register whenever an event is consumed. This register must be initialized to 0.

2.1.1.4.9 Command Consumer Index

The Command Consumer Index register contains the index of the next command in the Command Ring that will be consumed. The NIC writes this register whenever a command is consumed. This register must be initialized to 0.

2.1.1.4.10 Tuning Parameters

These are the tuning parameters that may be modified by the Host at any time to tune the NIC firmware.

Table 18. Tuning Parameters

Tuning Parameters	Offset
Receive coalesced ticks	0x00
Send coalesced ticks	0x04
Stat ticks	0x08
Send max coalesced BDs	0x0C
Receive max coalesced BDs	0x10
NIC tracing	0x14
Gigabit Link Negotiation	0x18
10/100 Link Negotiation	0x1c

2.1.1.4.10.1 Receive Coalesced Ticks

The Receive Coalesced Ticks register contains the number of clock ticks (of 1 microseconds each) that must elapse before the NIC DMAs the receive return producer pointer to the Host and generates an interrupt. This tunable parameter works in conjunction with the Receive Max Coalesced BDs tunable parameter. This NIC will return the receive return producer pointer to the Host when either of the thresholds is exceeded. A value of 0 means that this parameter is ignored and Receive BDs will only be returned when the Receive Max Coalesced BDs value is reached. This register must be initialized by the Host.

2.1.1.4.10.2 Send Coalesced Ticks

The Send Coalesced Ticks register contains the number of clock ticks (of 1 microseconds each) that must elapse before the NIC DMAs the send consumer pointer to the Host and generates an interrupt. This tunable parameter works in conjunction with the Send Max Coalesced BDs tunable parameter. This NIC will return the send consumer pointer to the Host when either of the thresholds is exceeded. A value of 0 means that this parameter is ignored and Send BDs will only be returned when the Send Max Coalesced BDs value is reached. This register must be initialized by the Host. For hosts that need to recover their send buffers this value should be set a value that will allow the Host to recover their send buffers in a reasonable time period. For hosts that do not need to recover their send buffers this value should be set to zero.

2.1.1.4.10.3 Stat Ticks

The Stat Ticks register contains the number of clock ticks (of 1 microseconds each) that must elapse before the NIC DMAs the statistics block to the Host and generates a STATS_UPDATED event. If set to zero then statistics are never DMAed to the Host. This register must be initialized by the Host. It is recommended that this value be set to a high enough frequency to not mislead someone reading statistics refreshes. Several times a second is enough.

NOTE: *The one microsecond clock tick referenced above is a nominal time and the actual hardware may not provide granularity to this level. For example, on Tigon 2 (Revision 6) cards with release 12.0 the clock granularity is 5 microseconds.*

2.1.1.4.10.4 Send Max Coalesced BDs

The Send Max Coalesced BDs register contains the number of buffer descriptors that will be coalesced before the NIC updates the Send Consumer Index. This register is set to 0 to disable send buffer descriptor coalescing. This register must be initialized by the Host. There is no simple recommendation that can be given for this register. The factors that affect this register's setting are size of the Send Ring and the Host's usage of send buffers.

2.1.1.4.10.5 Receive Max Coalesced BDs

The Receive Max Coalesced BDs register contains the number of buffer descriptors that will be coalesced before the NIC updates the Receive Return Ring Producer Index. This register is set to 0 to disable receive buffer descriptor coalescing. This register must be initialized by the Host. There is no simple recommendation that can be given for this register. The factors that affect this register's setting are the speed of the processor, the upper layers of the stack, and the interrupt processing time.

2.1.1.4.10.6 NIC Tracing

The NIC tracing register is a bit field that controls what runtime tracing is enabled.

NOTE: *The production code has the tracing disabled for performance reasons, so the NIC Trace Pointer, the NIC Trace Start, and the NIC Trace Length fields are invalid. A version of the firmware with tracing enabled is available in the Open Driver Kit.*

The meanings of the various bits are:

Table 19. NIC Tracing definitions

Bit	Name	Description
0x00000001	TRACE_TYPE_SEND	Enable all send related level 1 tracing.
0x00000002	TRACE_TYPE_RECV	Enable all recv related level 1 tracing.
0x00000004	TRACE_TYPE_DMA	Enable all DMA related level 1 tracing.
0x00000008	TRACE_TYPE_EVENT	Enable all event generation related level 1 tracing.
0x00000010	TRACE_TYPE_COMMAND	Enable all command processing related level 1 tracing.
0x00000020	TRACE_TYPE_MAC	Enable all MAC related level 1 tracing.
0x00000040	TRACE_TYPE_STATS	Enable all statistics related level 1 tracing.
0x00000080	TRACE_TYPE_TIMER	Enable all timer related level 1 tracing.
0x00000100	TRACE_TYPE_DISP	Enable all dispatcher related level 1 tracing.
0x00000200	TRACE_TYPE_MAILBOX	Enable all mailbox related level 1 tracing.

Table 19. NIC Tracing definitions (continued)

Bit	Name	Description
0x00000400	TRACE_TYPE_RECV_BD	Enable all receive buffer descriptor level 1 tracing.
0x00000800	TRACE_TYPE_LNK_PHY	Enable all Physical Layer of Link bringup level 1 tracing.
0x00001000	TRACE_TYPE_LNK_NEG	Enable all Link Negotiation level 1 tracing.
0x10000000	TRACE_LEVEL_1	Enable all level 1 tracing.
0x20000000	TRACE_LEVEL_2	Enable level 2 (data) tracing.

2.1.1.4.10.7 Gigabit Link Negotiation

The following table indicates the possible bit selection for the Link Negotiation parameter. You must set LNK_1000MB and LINK_FULL_DUPLEX. LNK_TX_FLOW_CTL_Y is only available on Tigon 2 ASICs. A normal setting for link negotiation enabled and receive flow control packets would be LNK_1000MB | LNK_FULL_DUPLEX | LNK_RX_FLOW_CTL_Y | LNK_NEG_FCTL | LNK_NEGOTIATE | LNK_ENABLE.

Table 20. Link Negotiation Bit Definitions

Bit	Name	Description
0x00002000	LNK_SENSE_NO_NEG	Link brought up without link negotiation
0x00004000	LNK_LOOPBACK	Set this link into loopback.
0x00008000	LNK_PREF	This link is preferred. May be set in only one of
0x00040000	LNK_1000MB	1000 megabit data rate
0x00080000	LNK_FULL_DUPLEX	set full duplex
0x00200000	LNK_TX_FLOW_CTL_Y	emit flow control packets (Tigon 2 only)
0x00800000	LNK_RX_FLOW_CTL_Y	obey received flow control packets
0x20000000	LNK_NEGOTIATE	enable autonegotiation with autosensing
0x40000000	LNK_ENABLE	enable link

Autosense is enabled if the LNK_NEGOTIATE bit is set. Autosense allows the NIC to automatically detect the way the connected hardware is attempting to bring up the link and act accordingly. Disabling LNK_NEGOTIATE disables autosensing as well.

2.1.1.4.10.8 10/100 Link Negotiation

The following table indicates the possible bit selection for the Link Negotiation parameter. You can either have the system autonegotiate the link speed or not by setting the LNK_NEGOTIATE bit. If this bit is set then it will autonegotiate depending on which of the LNK_10MB, LNK_100MB, LNK_FULL_DUPLEX, and LNK_HALF_DUPLEX bits you have set. That is you can limit the range of the negotiation by only setting some of these bits or you can allow any combination of these bits by setting them all. To hard code a speed or duplex setting just set the field appropriately without setting the LNK_NEGOTIATE bit.

LNK_TX_FLOW_CTL_Y is only available on Tigon 2 ASICs. Flow control will follow exactly the bit settings specified (as long as the link is in full duplex mode). That is, there is no negotiation of flow control on 10/100 links. A normal setting for link negotiation enabled, allowing any speed and duplex setting would be LNK_100MB | LNK_10_MB | LNK_FULL_DUPLEX | LNK_HALF_DUPLEX | LNK_NEGOTIATE | LNK_ENABLE.

Table 21. Link Negotiation Bit Definitions

Bit	Name	Description
0x00004000	LNK_LOOPBACK	Set this link into loopback.
0x00008000	LNK_PREF	This link is preferred. May be set in only one of
0x00010000	LNK_10MB	10 megabit data rate
0x00020000	LNK_100MB	100 megabit data rate
0x00040000	LNK_1000MB	1000 megabit data rate
0x00080000	LNK_FULL_DUPLEX	set full duplex
0x00100000	LNK_HALF_DUPLEX	set half duplex
0x00200000	LNK_TX_FLOW_CTL_Y	emit flow control packets (Tigon 2 only)
0x00800000	LNK_RX_FLOW_CTL_Y	obey received flow control packets
0x20000000	LNK_NEGOTIATE	enable autonegotiation (See the proposed 802.3z specification)
0x40000000	LNK_ENABLE	enable link

2.1.1.4.11 NIC Trace Pointer

The NIC Trace Pointer register contains a pointer to the current entry in the NIC trace buffer in NIC memory space. This register is initialized to the beginning of the NIC trace buffer by the NIC during initialization and is updated by the NIC as the trace buffer is used.

NOTE: *Tracing is compiled out in operational firmware, so the NIC Trace Pointer, the NIC Trace Start, and the NIC Trace Length fields are invalid. A version of the firmware with tracing enabled is included in the Open Driver Kit.*

2.1.1.4.12 NIC Trace Start

The NIC Trace Start register contains a pointer to the start of the NIC trace buffer in NIC memory space.

2.1.1.4.13 NIC Trace Length

The NIC Trace Length register contains the total length (in bytes) of the NIC Trace buffer in NIC memory space.

2.1.1.4.14 ifIndex

ifIndex is the Interface Index number assigned to this interface by the Host. This variable is initialized by the Host and the NIC firmware copies this into its stats area that is then DMAed to the Host. As per RFC-1213, this is a unique value, greater than zero.

2.1.1.4.15 ifMtu

ifMtu specifies the size of the largest packet which will be sent/ received on this interface, specified in octets. This value is initialized by the Host (and currently is never changed). The NIC firmware copies this into its stats area which is then DMAed to the Host. The value may not exceed 9018 if Jumbo Frames are enabled. It may not exceed 1518 if Jumbo Frames are not enabled.

This value should be set to the size of the largest Ethernet frame that you want to receive minus the CRC. The NIC uses this value to filter incoming frames as “too large”. If a frame is received that is larger than the ifMtu value it will be discarded and the dot3StatsFrameTooLongs counter will be incremented.

2.1.1.4.16 Mask Interrupts

This register acts similar to Mailbox 0. If it is non-zero, no interrupts will be generated. The difference is that it does not deassert the interrupt line, which touching Mailbox 0 will do. This feature is useful for operating systems that require the ability to mask interrupts at the chip. This is only necessary on the original Tigon ASIC. On Tigon 2 ASICs (of Revision 6) you should use the Mask Interrupt Bit in the Miscellaneous Host Control Register (See “Tigon Miscellaneous Host Control Register”, section 2.1.1.2.1). If using the hardware bit pl,ease ensure that the Mask Interrupts Register is 0 because the firmware still honors it.

2.1.1.4.17 Gigabit Link State

This register is valid after a Link State Event has been received. It uses the same set of bits as the Gigabit Link Negotiation field (See “Gigabit Link Negotiation”, section 2.1.1.4.10.7). This field indicates how the Gigabit Link State was negotiated. If the Link is down (according to the latest Link State Event) this register is invalid. This is a read-only register.

2.1.1.4.18 10/100 Link State

This register is valid after a Link State Event has been received. It uses the same set of bits as the 10/100 Link Negotiation field (See “10/100 Link Negotiation”, section 2.1.1.4.10.8). This field indicates how the 10/100 Link State was negotiated. If the Link is down (according to the latest Link State Event) this register is invalid. This is a read-only register.

2.1.1.4.19 Receive Return Consumer Index (OBSOLETE on Tigon 2)

The Receive Return Consumer Index register contains the index of the next Receive BD in the Receive Return Ring that will be consumed. The Host writes this register whenever a BD is consumed from the Receiver Return Ring. This register must be initialized to 0. Because it is impossible for the NIC to overrun this ring, this value is now ignored on Tigon 2. It is still required on Tigon 1.

2.1.1.4.20 Command Ring

The Command Ring is the 64 word Command Ring. (See “Command Ring”, section 2.4.1.1)

2.1.1.5 Local Memory Window

The Local Memory Window is a 2Kbyte region which provides the ability for the host to map any Tigon local memory region into the Shared Memory region.

This window is accessed by setting the Window Base Address register. (See “Tigon Window Base Address Register”, section 2.1.1.2.5.)

The Host uses this window to download the firmware into the NIC and to update the Send Ring. (See “Send Ring”, section 2.4.2.1)

2.1.2 General Information Block

The general information block lives in Host memory and contains the statistics area and the control blocks for the shared ring structures. The NIC DMAs this block from the Host during firmware initialization.

Table 22. General Information Block

General Information Block	Offset
Statistics	0x000 - 0x3FF
Event Ring Control Block	0x400 - 0x40F
Command Ring Control Block	0x410 - 0x41F
Send Ring Control Block	0x420 - 0x42F
Receive Standard Ring Control Block	0x430 - 0x43F
Receive Jumbo Ring Control Block	0x440 - 0x44F
Receive Mini RingControl Block	0x450 - 0x45F
Receive Return Ring Control Block	0x460 - 0x46F
Event Producer Pointer	0x470 - 0x477
Receive Return Ring Producer Pointer	0x478 - 0x47F
Send Consumer Pointer	0x480 - 0x487
Refresh Stats Pointer	0x488 - 0x48F

2.1.2.1 Statistics

The statistics are maintained by the NIC and transferred to Host memory on demand. All counters are cleared when a NIC reset occurs, or when a CLEAR_STATS command is received.

All statistics are four bytes long unless specified differently.

2.1.2.1.1 MAC Statistics

MIB Statistics, taken from RFC 1643, Ethernet-like MIB [7].

dot3StatsAlignmentErrors
dot3StatsFCSErrors
dot3StatsSingleCollisionFrames
dot3StatsMultipleCollisionFrames
dot3StatsSQETestErrors
dot3StatsDeferredTransmissions
dot3StatsLateCollisions
dot3StatsExcessiveCollisions
dot3StatsInternalMacTransmitErrors
dot3StatsCarrierSenseErrors
dot3StatsFrameTooLongs
dot3StatsInternalMacReceiveErrors

2.1.2.1.2 Interface Statistics

MIB Statistics, taken from RFC 1213 and RFC 1573, MIB-II [8, 9], interfaces group.

ifIndex
ifType
ifMtu
ifSpeed
ifAdminStatus
ifOperStatus
ifLastChange
ifInDiscards
ifInErrors
ifInUnknownProtos
ifOutDiscards
ifOutErrors
ifOutQLen
ifPhysAddress (8 bytes)
ifDescr (32 bytes)
ifHCInOctets (8 bytes)
ifHCInUcastPkts (8 bytes)
ifHCInMulticastPkts (8 bytes)
ifHCInBroadcastPkts (8 bytes)
ifHCOutOctets (8 bytes)
ifHCOutUcastPkts (8 bytes)
ifHCOutMulticastPkts (8 bytes)
ifHCOutBroadcastPkts (8 bytes)
ifLinkUpDownTrapEnable
ifHighSpeed
ifPromiscuousMode
ifConnectorPresent

2.1.2.1.3 Alteon Networks Proprietary MIB Statistics for the NIC

2.1.2.1.3.1 Host Commands Statistics

nicCmdsSetHostState - Number of TG_CMD_HOST_STATE commands received. For the current and following commands, Refer to section 3.2 for the command descriptions.

nicCmdsFDRFiltering - Number of TG_CMD_FDR_FILTERING commands received

nicCmdsSetRecvProdIndex - Number of TG_CMD_SET_RECV_PRODUCER_INDEX commands received

nicCmdsUpdateGencommStats - Number of TG_CMD_UPDATE_GENCOM_STATS commands received

nicCmdsResetJumboRing - Number of TG_CMD_RESET_JUMBO_RING commands received

nicCmdsAddMCastAddr - Number of TG_CMD_ADD_MULTICAST_ADDR commands received

nicCmdsDelMCastAddr - Number of TG_CMD_DEL_MULTICAST_ADDR commands received

nicCmdsSetPromiscMode - Number of TG_CMD_SET_PROMISCUOUS_MODE commands received

nicCmdsLinkNegotiate - Number of TG_CMD_LINK_NEGOTIATION commands received

nicCmdsSetMACAddr - Number of TG_CMD_SET_MAC_ADDR commands received

nicCmdsClearProfilee - Number of TG_CMD_CLEAR_PROFILE commands received

nicCmdsSetMulticastMode - Number of TG_CMD_SET_MULTICAST_MODE commands received

nicCmdsClearStats - Number of TG_CMD_CLEAR_STATS commands received

nicCmdsSetRecvJumboProdIndex - Number of TG_CMD_SET_RECV_JUMBO_PRODUCER_INDEX commands received

nicCmdsSetRecvMiniProdIndex - Number of TG_CMD_SET_RECV_MINI_PRODUCER_INDEX commands received

nicCmdsRefreshStats - Number of TG_CMD_REFRESH_STATS commands received

nicCmdsUnknown - Number of unknown or illegal commands received

2.1.2.1.3.2 NIC Events Statistics

nicEventsNICFirmwareOperational- Number of TG_EVENT_FIRMWARE_OPERATIONAL events generated.

nicEventsStatsUpdated - Number of TG_EVENT_STATS_UPDATED events generated.

nicEventsLinkStateChanged - Number of TG_EVENT_LINK_STATE_CHANGED events generated.

nicEventsError - Number of TG_EVENT_ERROR events generated

nicEventsMCastListUpdated - Number of TG_EVENT_MULTICAST_LIST_UPDATED events generated.

nicEventsResetJumboRing - Number of TG_EVENT_RESET_JUMBO_RING commands received.

2.1.2.1.3.3 NIC Ring Manipulation Statistics

nicRingSetSendProdIndex - Number of times the NIC has seen updates to the Send Producer Index (See “Send Consumer Pointer”, section 2.1.2.5)

nicRingSetSendConsIndex - Number of times the send consumer index was updated

nicRingSetRecvReturnProdIndex - Number of times the recv return producer index was updated

2.1.2.1.3.4 Host Interrupts

nicInterrupts - Number of interrupts generated by NIC

nicAvoidedInterrupts - Number of interrupts avoided by NIC

2.1.2.1.3.5 NIC BD Coalescing Threshholds

nicEventThreshholdHit - Number of times Event Max Coalesce BD Threshold hit

nicSendThreshholdHit - Number of times either BD_FLAG_COAL_NOW or Send Max Coalesce BD Threshold hit.When these two conditions meet, the firmware sends updated consumer index to the host.

nicRecvThreshholdHit - Number of times Recv Max Coalesce BD Threshold hit. When this happens, the firmware updates the Receive Return Producer Index in the host.

2.1.2.1.3.6 DMA Attentions

nicDmaRdOverrun - Number of times DMA read overrun error occurred. This DMA error and following DMA errors are detected and reported by hardware. This error indicates that DMA FIFO data is overwritten by the new data from DMA read. If this counter is non-zero it indicates a serious error.

nicDmaRdUnderrun - Number of times DMA read underrun error occurred. This error indicates that DMA read engine is trying to read from an empty DMA FIFO. If this counter is non-zero it indicates a serious error.

nicDmaWrOverrun - Number of times DMA write overrun error occurred. This error indicates that DMA FIFO data is overwritten by the new data from DMA write. If this counter is non-zero it indicates a serious error.

nicDmaWrUnderrun - Number of times DMA write underrun error occurred. This error indicates that DMA write is rying to retrieve data from an empty DMA FIFO. If this counter is non-zero it indicates a serious error.

nicDmaRdMasterAborts - Number of times DMA read Master Abort error occurred. Possible cause is hardware failure or a bad address given to the NIC by the host driver

nicDmaWrMasterAborts - Number of times DMA write master Abort error occurred. Possible cause is hardware failure or a bad address given to the NIC by the host driver

2.1.2.1.3.7 Ring/ Miscellaneous Resources

nicDmaWriteRingFull - Number of times DMA write ring was full. The NIC firmware was unable to queue the data being sent from the NIC to the host in the DMA write ring. This indicates

data moving to the host is slower than write requests generated by the NIC, typically due to a slow PCI bus/ bridge chip

nicDmaReadRingFull - Number of times DMA read ring was full. The NIC firmware was unable to queue the data being sent from the host to the NIC in the DMA read ring. This indicates data moving to NIC is slower than read requests generated by the NIC, typically due to a slow PCI bus/ bridge chip but sometimes due to a very large burst of data from the host.

nicEventRingFull - Number of times NIC Event ring was full. This indicates events are being generated faster than they are processed.

nicEventProducerRingFull - Number of times the DMA write ring was full trying to place either the DMA event or event producer to host. This indicates events are generated faster than they can be processed.

nicTxMacDescrRingFull - Number of times MAC TX Descriptor ring was full. Possible cause is not enough TX buffer space

nicOutOfTxBufSpaceFrameRetry - Number of times TX buffer space was full. Possible cause is not enough TX buffer space

nicNoMoreWrDMADescriptors - Number of times NIC ran out of DMA write Descriptors. This indicates the DMA can not keep up with write requests. This is accompanied by a nicDmaWriteRingFull.

nicNoMoreRxBDs - Number of times NIC ran out of the Recv Buffer Descriptors. This indicates that the host is not replenishing empty receive BDs quickly enough

nicNoSpaceInReturnRing - Number of times NIC could not place a buffer descriptor in the return ring because it was full. Possible cause is the host is too slow in processing packets.

nicSendBDs - Number of Send Buffer Descriptors current being handled in the NIC (host based send rings only)

nicRecvBDs - Number of Standard Receive Buffer Descriptors available to the NIC.

nicJumboRecvBDs - Number of Jumbo Receive Buffer Descriptors available to the NIC.

nicMiniRecvBDs - Number of Mini Receive Buffer Descriptors available to the NIC (valid in Tigon 2 only).

nicTotalRecvBDs - Total number of Receive Buffer Descriptors available to the NIC.

nicTotalSendBDs - obsolete counter

nicJumboSpillOver - Number of times a Jumbo frame was split into multiple standard BDs.

nicSbusHangCleared - Number of times an SBus DMA bug was worked around.

nicEnqEventDelayed - Number of times posting an event was delayed.

2.1.2.1.3.8 Internal MAC RX Statistics

nicMacRxLateColls - Packets dropped due to late collisions. This error does not happen at Gigabit speed as the NIC operates in full duplex mode.

nicMacRxLinkLostDuringPkt - Packets dropped because of loss of link. Possible causes are cable disconnected, broken cable, NIC hardware failure, or the link partner hardware failure.

nicMacRxPhyDecodeError - Packets dropped because of PHY decode errors. Possible cause is hardware failure which generates too much noise such that the PHY can not recognize the signal. Another possible cause is hardware failure.

nicMacRxMacAbort - Packets aborted by MAC because of remote errors. Possible causes are receiving a packet smaller than minimal size (64 bytes), a PHY decode error (see above), a collision was detected during receipt, or an error occured during gigabit half duplex frame extension.

nicMacTruncNoResources - Number of times a frame was dropped due to lack of NIC internal resources (e.g. memory space or MAC descriptors). This is usually because the bus is too slow. Increase RX buffer ratio, live with it, or invoke flow control.

nicMacRxDropUla - Unicast Packets dropped; this is caused by a nonmatching unicast address. This happens regularly on full duplex repeaters or during switch flooding.

nicMacRxDropMcast - Multicast Packets dropped; destination address in the received packet doesn't match the Multicast address list. This is normal for multicast packets for groups not set up by the host.

nicMacRxFlowControl - Number of Flow Control packets received.

nicMacRxDropSpace - Packets dropped because of lack of space. This is usually because the bus is too slow. Increase RX buffer ratio, live with it, or invoke flow control.

nicMaxRxColls - Total number of packets dropped because of collisions. This is caused by two nodes sending messages simultaneously. This error does not happen at Gigabit speeds as the NIC is operating in full duplex mode.

nicMacRxTotalAttentions - Total number of Mac Rx Attentions, e.g. Receive Descriptor Attention, Receive Buffer Attention, Flow Control XON Sent, Flow Control XOFF Sent, and FIFO overrun.

nicMacRxLinkAttentions - Total number of link state change Attentions, e.g. Auto-Negotiation changed, Link State Error, and Link Ready Changed.

nicMacRxSyncAttentions - Total number of Sync lost Attentions.

nicMacRxConfigAttentions - Total number of Link Config Attentions. Possible causes are the link partner changing its link configuration. Under normal situation, this attention does not indicate hardware failure, only when this number is greater than the number of times link partner configuration changed.

nicMacReset - Total number of times MAC was reset. This is caused by link lost and trying to re-establish the link.

nicMacRxBufDescrAttns - Obsolete

nicMacRxBufAttns - Total number of Receive Buffer Descriptor Attentions. This attention is triggered by received buffer descriptor reaches a predefined threashold. This happens when the receive buffer fills either due to a slow host or a bad frame that has to be manually removed by the firmware.

nicMacRxZeroFrameCleanup - Number of times Received Buffer got cleaned up when the Receive Buffer is full and the frame received count is 0. This indicates the data in the receive buffer is all garbage. This could happen if the remote part kept sending errored frames.

nicMacRxOneFrameCleanup - Number of times Received Buffer got cleaned up when the Receive Buffer is full and the frame received count is 1. This could happen if the remote part kept sending errored frames.

nicMacRxMultipleFrameCleanup - Number of times Received Buffer got cleaned up when the Receive Buffer is full and the frame received count is more than one. This could happen if the remote part kept sending errored frames.

nicMacRxTimerCleanup - Number of times setting up a timer to wait for Receive Buffer space to be freed up when the Receive Buffer is full. This could happen if the remote part kept sending errored frames.

nicMacRxDmaCleanup - Total Number of times DMA Buffer got cleaned up when the Receive Buffer is full. This could happen if the remote part kept sending errored frames.

2.1.2.1.3.9 Internal MAC TX Statistics

nicMacTxCollision[1] - Tx collision histogram

nicMacTxCollision[2]

nicMacTxCollision[3]

nicMacTxCollision[4]

nicMacTxCollision[5]

nicMacTxCollision[6]

nicMacTxCollision[7]

nicMacTxCollision[8]

nicMacTxCollision[9]

nicMacTxCollision[10]

nicMacTxCollision[11]

nicMacTxCollision[12]
nicMacTxCollision[13]
nicMacTxCollision[14]
nicMacTxCollision[15]
nicMacTxTotalAttentions - Total number of Tx Attentions
nicProfile[32] - NIC profiling is not enabled, ignore these statistics

2.1.2.2 Ring Control Blocks

Each shared ring (See “Shared Rings”, section 2.4) has a Ring Control Block (RCB) associated with it. An RCB has the following format.

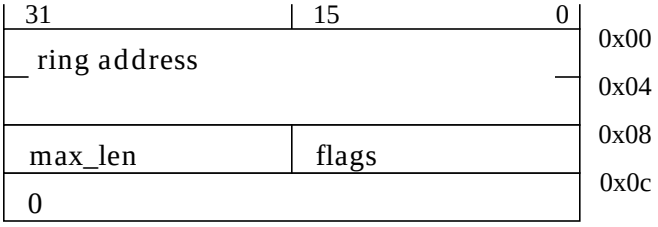


Figure 2. Ring Control Block

The ring address is the Host address of the first ring element. The ring address is in Host address format (See “Host Addresses”, section 1.3.1). Two of the rings have addresses in the NIC space (in particular, the Command and Send Rings). The address in this case is a NIC address, but it is still in Host address format (See Figure 1).

Flags contains the flag bits for Send and Receive Rings. The feature is active when the bit is set.

Table 23. RCB Flags

Bit	Name	Description
0x00000001	RCB_FLAG_TCP_UDP_CKSUM	Perform TCP or UDP checksum calculations.
0x00000002	RCB_FLAG_IP_CKSUM	Perform IP checksum calculations.
0x00000008	RCB_FLAG_NO_PSEUDO_HDR_CKSUM	Do not include the pseudo header in TCP or UDP checksum calculations (Only valid if RCB_FLAG_TCP_UDP_CKSUM flag is set). In order to get the correct checksum the driver must seed the TCP/UDP checksum field with the pseudo header’s checksum.
0x00000010	RCB_FLAG_VLAN_ASSIST	Enable VLAN tagging insertion or deletion.
0x00000020	RCB_FLAG_COAL_UPDATE_ONLY	When this bit is set the NIC will only generate interrupts for transmit completions when the Send Coallesced Ticks value is exceeded. Send Consumer updates are still made based on the Send Max Coallesced BDs and the BD_FLAG_COAL_NOW.
0x00000040	RCB_FLAG_HOST_RING	For Send RCB only. Indicates the send ring is located in host memory, not in NIC memory. The ring address is a bus virtual or physical address in the host.

Table 23. RCB Flags

Bit	Name	Description
0x00000080	RCB_FLAG_IEEE_SNAP_CKSUM	NIC will parse 802.3 SNAP frames and do checksum offload on them
0x00000100	RCB_FLAG_USE_EXT_RECV_BD	For Jumbo Recv RCB only. Indicates that the Jumbo Recv Ring will use extended receive buffer descriptors.(See Figure 11.)
0x00000200	RCB_FLAG_RING_DISABLED	For Jumbo Recv or Mini Recv RCB only. Indicates Ring will never be used. If a Jumbo or Mini Ring mailbox is updated (or the equivalent command is sent) the NIC will halt.

NOTE: *In order to use the checksum offload feature, the RCB_FLAG_TCP_UDP_CKSUM and/or the RCB_FLAG_IP_CKSUM bit must be set prior to starting the firmware.*

The Send Ring RCB uses the max_len field to determine the total number of entries in the send ring (see extended Send Ring Handling below).

The Receive Standard Ring RCB uses the max_len field to indicate the length of each buffer placed into the standard ring. This way any frame received that is larger than this value will attempt to use a jumbo ring buffer instead of a standard ring buffer.

The Receive Mini Ring RCB uses the max_len field to indicate the length of each buffer placed into the mini ring. This way any frame received that is less than or equal to this value will be placed in the mini ring if buffer descriptors are available.

The Receive Return Ring RCB uses the max_len field to indicate the number of entries in the ring. The only valid sizes are 2048, 1024, and 0 (which is equivalent to 2048 for backward compatibility). For 1024 to be a valid size the Receive Mini Ring RCB flag field must have the RCB_FLAG_RING_DISABLED bit set. If an invalid size is set the NIC will halt with an error during initialization.

The max_len field is unused in the Event, Command, and Receive Jumbo Ring Control Blocks.

2.1.2.3 Event Producer Pointer

The Event Producer Pointer is a Host Address which points to the location where the NIC places the Event Producer Index. The Event Producer Pointer is in Host address format (See “Host Addresses”, section 1.3.1) and points to an aligned eight byte region where only the first four bytes contain relevant information. The remaining four bytes are not explicitly set to zero, i.e. they may contain an unknown value and must be ignored by the Host.

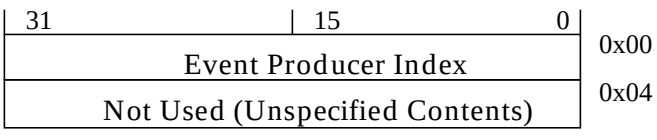


Figure 3. Event Producer

2.1.2.4 Receive Return Ring Producer Pointer

Receive Return Ring Producer Pointer is a Host Address which points to the location where the NIC places the Receive Return Ring Producer Index. The Receive Return Ring Producer Pointer is in Host address format (See “Host Addresses”, section 1.3.1) and points to an aligned eight byte region where only the first four bytes contain relevant information. The remaining four bytes are not explicitly set to zero, i.e. they may contain an unknown value and must be ignored by the Host.

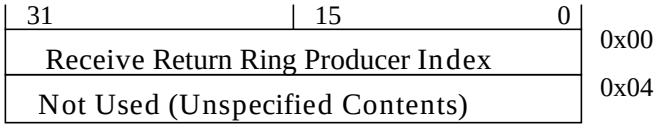


Figure 4. Receive Return Ring Producer

2.1.2.5 Send Consumer Pointer

The Send Consumer Pointer is a Host Address which points to the location where the NIC places the Send Consumer Index. The Send Consumer Pointer is in Host address format (See “Host Addresses”, section 1.3.1) and points to an aligned eight byte region where only the first four bytes contain relevant information. The remaining four bytes are not explicitly set to zero, i.e. they may contain an unknown value and must be ignored by the Host.

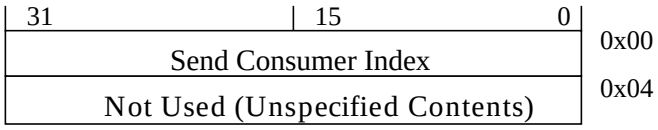


Figure 5. Send Consumer

2.1.2.6 Refresh Stats Pointer

The Refresh Stats Pointer is a Host Address which points to the location where a new set of statistics can be placed which can then be loaded into the NIC. This is useful if your system supports a hot swap capability on its bus. This way a new NIC card can be placed into the system and, with proper driver support, new statistics can be updated with the TG_CMD_REFRESH_STATS command to allow this card to transparently take over the function of a failed card, even to the point that the statistics are consistent. The Refresh Stats Pointer is in Host address format (See “Host Addresses”, section 1.3.1) and points to a region which is identical in structure to the statistics region (See “Statistics”, section 2.1.2.1)

2.2 SBus Memory Model

The SBus uses a single linear address space. Different sections of the address space are used to access different sections of the SBus card and the Tigon ASIC.

Table 24. SBus Address Space

Address Range	Access	Region
0x00000000 - 0x0000FFFF	Byte R/O	FCode PROM
0x00010000 - 0x0001003C	32-bit R/W	PCI Configuration Region
0x00010040 - 0x0001FFFF	N/A	Undefined Region
0x00020000 - 0x00023FFF	32-bit R/W	Shared Memory Region
0x00024000 - 0xFFFFFFFF	N/A	Undefined Region

2.3 Serial EEPROM

The serial EEPROM contains manufacturing information and initialization code as described in Chapter 5 of this document.

2.3.1 Reading and Writing the Serial EEPROM

To read or write the serial EEPROM requires a fairly complex algorithm. It is strongly recommended that you port the code from the sample driver. If you choose to rewrite this code, the serial EEPROM is an I^2C device. In particular, it is an Atmel AT24C64 device. The device address used is 010000[R/W] (binary). For read set the R/W bit to 1, for write set the R/W bit to 0.

2.3.2 Data fields in the Serial EEPROM

2.3.2.1 MAC Address

The MAC address is an 8 byte field with the bottom 6 bytes being the actual Ethernet MAC address. The top two bytes are 0 (See “MAC Addresses”, section 1.3.3). This field is located at offset 0x8c in the Serial EEPROM.

2.3.2.2 Software Key

There is a software key field in the EEPROM that allows the driver firmware to allow or disallow features depending on the values in these fields. The first character of this 16 byte field is the software vendor identifier. Your driver should check this value against the value given to you by Alteon Networks so that your customers don’t buy cards from other sources and attempt to use them with your driver. The software key field is located at offset 0x94 in the Serial EEPROM.

2.3.2.3 Customer Data Area

There is a 32 byte free form region that the customer can use for any purpose they want. This region is from offset 0xd4 - 0xf3.

2.4 Shared Rings

The Host and the NIC use a series of shared rings to communicate with each other. While each ring itself is shared, there are two indices that control the operation of each ring which are not shared. These are the producer index and the consumer index. The producer adds elements to the ring and increments the producer index while the consumer removes elements from the ring and increments the consumer index. When the producer and consumer indices are equal, the ring is empty. When the producer is one “behind” the consumer, the ring is full.

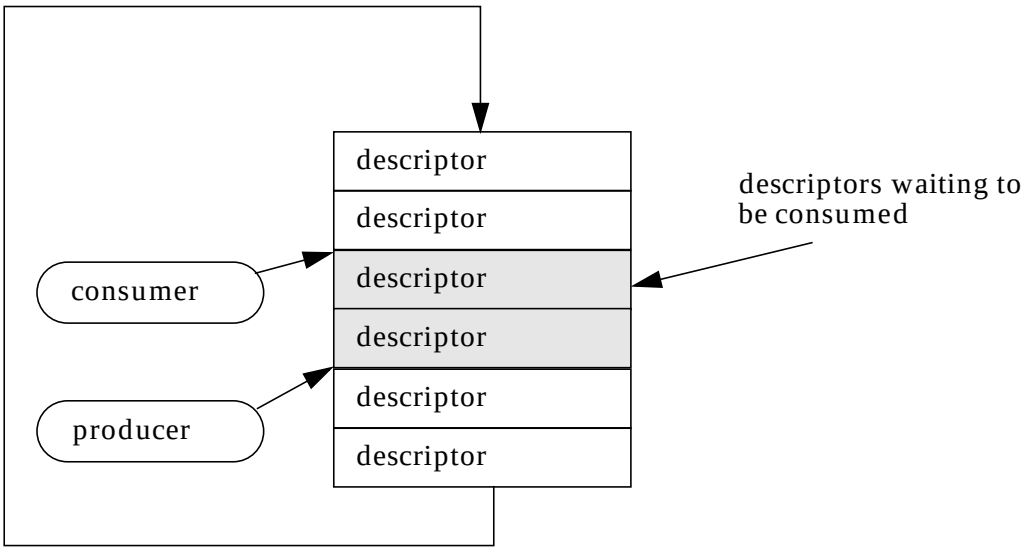


Figure 6. Producer-Consumer model

2.4.1 Control Rings

Control rings contain their entire information in the ring element. There are two control rings, the Command Ring and the Event Ring.

2.4.1.1 Command Ring

The Command Ring holds commands from the Host intended for the NIC. The Host controls the Command Ring producer and the NIC controls the Command Ring consumer. The ring element for the Command Ring is a command (See “Commands”, section 3.2). The Command Ring holds 64 command elements and lives in shared memory. The Command Producer lives in the shared memory and is a mailbox. The Command Consumer also lives in shared memory.

2.4.1.2 Event Ring

The Event Ring holds responses from the NIC to the Host. The NIC controls the Event Ring producer and the Host controls the Event Ring consumer. The ring element for the Event Ring is an event (See “Events”, section 3.3). The Event Ring holds 256 event elements and lives in host memory. The producer also lives in host memory and is pointed to by the Event Producer Pointer (See “Event Producer Pointer”, section 2.1.2.3). The consumer lives in shared memory.

When the Event Ring is full, the NIC stops generating events until space is available.

2.4.2 Data Rings

Data rings contain some information about the data in the ring element and a pointer to the actual data buffer. There are two sets of data rings, a single Send Ring and set of Receive Rings. The ring element for all data rings is the buffer descriptor (See “Buffer Descriptors”, section 3.1).

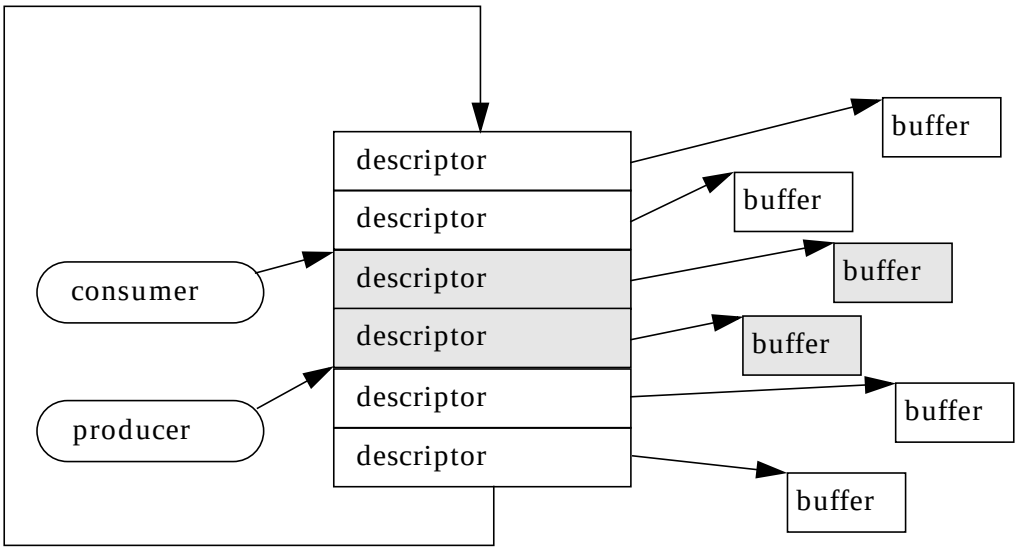


Figure 7. Producer-Consumer Model with data

2.4.2.1 Send Ring

The Send Ring is used to transfer data from the Host to the NIC for transmission onto the network. The Host is the Send Ring producer and the NIC is the Send Ring consumer. The Send Ring holds a variable number of buffer descriptors (See “Extended Send Ring Handling”, section 2.4.2.2).

This ring lives in NIC memory. The location of this ring, in NIC memory, is indicated by the Send Ring RCB (See figure 2 on page 41). There are limitations on this value as indicated in the Extended Send Ring Handling section below. The Host writes to the Send Ring by using the Local Memory Window (See “Local Memory Window”, section 2.1.1.5)

2.4.2.2 Extended Send Ring Handling

Prior to version 10.2 of this API document, the Send ring was at a fixed location (0x3800) and was of a fixed length (128 entries). This fixed approach was taken to simplify the operation of the Tigon base window register. The ring fit perfectly into the 2k region of the ring and the base could be set once to 0x3800 and left alone. Unfortunately, some situations occured that indicated that this ring size could be too small. To alleviate this, the ring size can now be specified in the max_len field of the Send RCB. The ring size can be any of three values, 128, 256, or 512 with respective starting addresses of 0x3800, 0x3000, and 0x2000. The ring fits into NIC memory between the locations 0x2000 and 0x3fff and always ends at 0x3fff. It is up to the driver to manipulate the Tigon Window base register in such a way that the Window is pointing into the proper section of the ring when writing Send Buffer Descriptors.

2.4.2.3 Receive Rings

A set of Receive Rings are used to transfer data that has been received from the network from the NIC to the Host. There are three rings into which the Host places BDs that point to empty buffers, and one ring that the NIC uses to place BDs that point to filled buffers. Three rings are used to allow for different sized data buffers. One ring, the Standard Ring, uses traditional (for Ethernet) 1514 byte buffers. Another ring, the Jumbo Ring, uses extended 9014 byte buffers. The last ring, the Mini Ring, uses small buffers whose maximum length can be specified in the max_len field of its Ring Control Block. On Tigon 2 products the Standard Ring contains 512 entries, the Jumbo Ring contains 256 entries, the optional Mini Ring contains 1024 entries and the Return Ring contains either 1024 or 2048 entries, depending on whether the Mini Receive Ring is enabled and the value placed in the max_len field of the Receive Return Ring's RCB. The Mini Receive Ring is only supported on Tigon 2 products. On Tigon 1 products the Standard Ring contains 512 entries, the Jumbo Ring contains 256 entries, and the Return Ring contains 2048 entries.

The Host writes the Receive Buffer Descriptors, pointing to empty buffers, into the Receive Standard Ring and Receive Jumbo Ring and hands them off to the NIC (via the TG_CMD_SET_RECV_PRODUCER_INDEX and the TG_CMD_SET_RECV_JUMBO_PRODUCER_INDEX commands). When the NIC fills the buffers associated with these Buffer Descriptors, it returns them back to the Host in the Receive Return Ring. The Host determines which buffer descriptor was used by looking at the flag bit (to indicate the Standard or Jumbo Ring) and index. (See figure 10 on page 57).

The Jumbo Ring is optional. If you do not send a TG_CMD_SET_RECV_JUMBO_PRODUCER_INDEX command (or update the Jumbo Receive Producer Mailbox on Tigon 2 NICs) then the NIC will not use the Receive Jumbo Ring.

Similarly, the Rings do not need to be filled beyond the number of buffers you wish to give to the NIC. If you want to trade off memory for performance, or you feel that your driver can keep buffers available for the NIC receive engine without using all 512 Standard entries or 256 Jumbo entries, just don't completely fill the ring using the TG_CMD_SET_RECV_PRODUCER_INDEX or TG_CMD_SET_SET_RECV_JUMBO_PRODUCER_INDEX commands (or the equivalent mailboxes on Tigon 2 NICs).

The Mini Ring is also optional. If you do not update the Mini Receive Producer Mailbox on then the NIC will not use the Mini Receive Ring. In all cases, if the frame will not fit in a Mini Receive ring or there are no Mini Receive Ring buffers available a Standard Ring buffer will be used instead.

Also, the Jumbo Receive Ring can be filled with Extended Receive Buffer Descriptors (See Figure 11) if the RCB_FLAG_USE_EXT_RECV_BD is set in the Jumbo RCB. This extended Receive Buffer Descriptor allows the host to describe large buffers that are not contiguous. Up to 4 non-contiguous areas can be supported.

If a large frame arrives and the NIC is out of Jumbo buffer descriptors and the DONT_FRAG_JUMBOS bit is not set in the Operating Mode register then the frame will be split into multiple standard buffer descriptors. Unlike Standard buffers, Jumbo and Mini buffer descriptors always contain one complete packet.

2.5 Interrupts

2.5.1 Interrupt Generation

Since interrupts to the Host are typically quite expensive, they are generated only if there is work for the Host. Any time a new event is generated by the NIC, for which the Host has not been notified, an Interrupt is generated. However, the "Interrupt Avoidance" mechanism may choose to suppress this interrupt and not post it to the Host depending upon the state of the Host (See "Interrupt Avoidance", section 2.5.2 for further clarification).

Generation of the events can be tuned by coalescing the send and receive events.

NOTE: *The word event is used generically in this section to refer to events posted to the event ring or the update of the Send Consumer or the Receive Return Ring Producer.*

The interrupt handler or, in the case of some OS's, the event handling routine, in Host software is expected to process all available events before exiting.

An interrupt may be cleared explicitly using the Miscellaneous Host Control Register (See "Tigon Miscellaneous Host Control Register", section 2.1.1.2.1) or implicitly by writing into Mailbox 0.

2.5.2 Interrupt Avoidance

The NIC will not interrupt the Host if the Host indicates that it is already processing events. When the Host is processing events, the NIC assumes that the Host will process *all* the pending events before returning from the "event processing" routine.

The NIC checks two conditions in order to determine whether the Host is presently processing events. These two conditions are:

- The Interrupt State Bit in the Tigon Miscellaneous Host Control Register.
- The Host-NIC handshake, whereby the Host explicitly indicates to the NIC that it is processing events and therefore does not want to take any interrupts. (See "Host In Interrupt Handler", section 2.1.1.3.1)

If either of the above two conditions is true, an Interrupt is not generated.

2.5.3 Masking Interrupts

For systems that need to mask interrupts at the card without changing the level of the interrupt line, this can be accomplished through the use of the Mask Interrupts field of the General Communications Area (See "General Communications Registers", section Table 14.) for the original Tigon ASIC or through the use of the Mask Interrupt bit of the Miscellaneous Host Control Register (See "Tigon Miscellaneous Host Control Register", section 2.1.1.2.1) for the Tigon 2 ASIC.

2.6 Checksum Offload

2.6.1 Preparing the NIC for Checksum Offload

In order to prepare the NIC to do checksum offload, you must set the proper bits in the send and receive RCBs (See “Ring Control Blocks”, section 2.1.2.2). There are three bits that control checksum offload, RCB_FLAG_TCP_UDP_CKSUM, RCB_FLAG_IP_CKSUM, and RCB_FLAG_NO_PSEUDO_HDR_CKSUM. The first two bits indicate what checksum offload capabilities are enabled in the firmware. The last bit indicates the algorithm used to calculate the checksum. These bits may be set differently for send and receive. However, make sure that all enabled receive rings have the same flags set or an ERROR event will be generated.

NOTE: *Do not set any capabilities that you do not wish to use. Setting extra bits changes the algorithms used by the firmware and can increase the latency and decrease the throughput of the NIC.*

2.6.2 Per Packet Settings When Using Checksum Offload

2.6.2.1 The Send Case

The NIC can calculate the IP, TCP, and/or UDP checksum and insert it into the outgoing frame. It can even generate the checksum for IP fragmented packets. To have the NIC generate the checksum include the appropriate bits in each buffer descriptor associated with a frame and either zero or seed the checksum fields that are to be calculated.

For example, suppose that the host wants to send a packet that consists of three separate pieces of data but is a single TCP frame. Further suppose that you want the NIC to calculate the IP and TCP checksums, but you don’t want the pseudo-header included in the TCP checksum calculation. First place a zero in the IP checksum field. Second, place the checksum of the pseudo-header in the TCP checksum field. Then generate the three buffer descriptors associated with the data. The first two buffer descriptor should set the flags BD_FLAG_TCP_UDP_CKSUM and IP_FLAG_IP_CKSUM. The last buffer descriptor should set the flags BD_FLAG_TCP_UDP_CKSUM, BD_FLAG_IP_CKSUM, BD_FLAG_END, and (for the send case not running checksum offload) one of the BD_FLAG_UCAST_PKT, BD_FLAG_BCAST_PKT, or BD_FLAG_MCAST_PKT flags.

2.6.2.2 The Send Case for IP Fragmented Packets

The NIC has the ability to calculate the TCP or UDP checksum for an IP fragmented packet if all of the IP fragments are placed into the send ring **consecutively and in the correct order**. To use this feature set the BD_FLAG_IP_FRAG_END bit in the very last buffer descriptor associated with the last fragment and set the BD_FLAG_IP_FRAG bit in each of the other buffer descriptors.

2.6.2.3 Limitations

Previous limitations on checksum offload (such as IP optioned packets and tagged VLAN frames) have been removed. These frames will be properly checksummed.

2.6.3 The Receive Case

The NIC can calculate the IP, TCP, and/or UDP checksum and forward the calculation to the host along with each incoming frame. The host can use these values directly for unfragmented packets or can offload the checksum calculation for IP fragmented packets. That is, the TCP/UDP checksum will include all the necessary data for unfragmented packets. For fragmented packets the host needs to add up the TCP/UDP checksums for allt the fragments to obtain the final checksum.

The NIC generates the checksum based on the bits set in the receive ring RCBs. It is important that all of the rings have the same bits set. If they do not, an ERROR event will be returned by the NIC at initialization time.

The checksums calculated by the NIC are placed into the buffer descriptors in the receive return ring. The TCP or UDP checksum will be calculated with the pseudo-header unless the RCB_FLAG_NO_PSEUDO_HDR_CKSUM field is placed in the receive ring RCBs.

2.7 VLAN Assist

The NIC has the ability to insert VLAN tags into transmitted frames and delete VLAN tags from received frames. The tags added and deleted are 802.1q compliant tags. An 802.1q compliant VLAN frame means that there is an Ethertype field of 0x8100 in the Ethertype/Length field of the the Ethernet header, there is a 16 bit VLAN tag field that is made up of 1 CNI bit, 3 priority bits, and 12 VLAN Identification bits, and the “real” 16 bit Ethertype/Length field follows the 16 bit VLAN tag field. Throughout the rest of this discussion the term “VLAN tag” is used to refer to the middle 16 bit field.

2.7.1 Tag Insertion for Outgoing Frames

To enable tag insertion you must specify that you may use it in the send RCB. Set flag RCB_FLAG_VLAN_ASSIST.

To send a frame with a tag inserted you must specify the sixteen bit tag value in the Send Buffer Descriptor (See “Send Buffer Descriptors”, section 3.1.2) and specify the buffer descriptor flag BD_FLAG_VLAN_TAG in the first buffer descriptor of a frame.

2.7.2 Tag Deletion for Incoming Frames

To enable tag deletion you must specify that you may use it in the receive RCBs. Set flag RCB_FLAG_VLAN_ASSIST in *all* the receive RCBs. If the RCB_FLAG_VLAN_ASSIST flag is set differently in different RCBs an Error Event will be generated and the NIC firmware will halt.

When a tagged frame is received by the NIC, the NIC will set the BD_FLAG_VLAN_TAG bit in the flags field of the buffer descriptor. The actual sixteen bit tag value will be placed in the appropriate field of that buffer descriptor (See “Receive Buffer Descriptors”, section 3.1.1).

2.8 Transmit Flow Diagram

The following diagram shows how a single frame is sent from the Host to the NIC and onto the network. For this example, the frame is described in a single buffer descriptor.

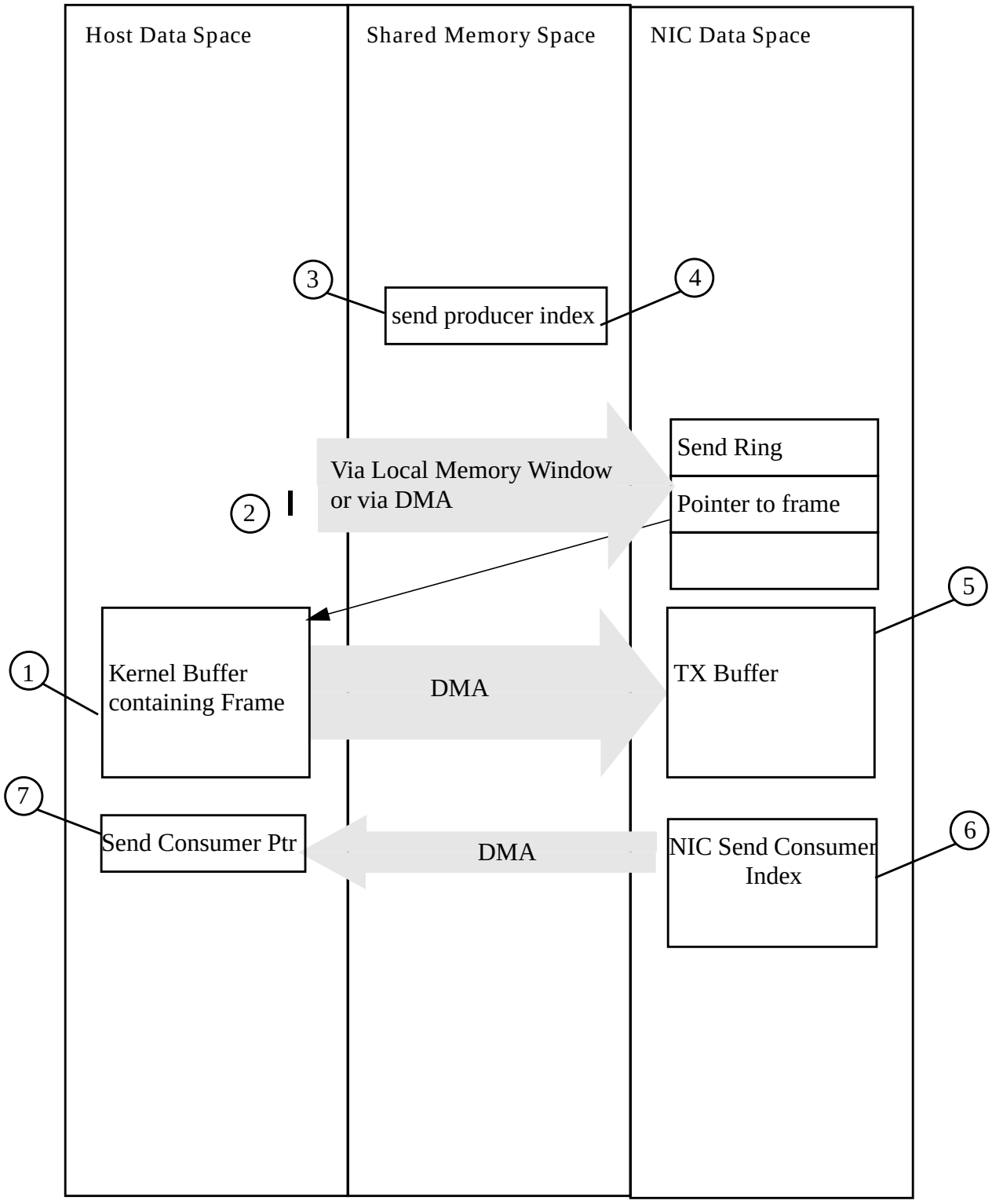


Figure 8. Transmit Flow Diagram

1. The Host creates a frame in Host memory.
2. The Host creates a buffer descriptor or series of buffer descriptors that describe the frame and places it or them in the Send Ring on the NIC, using the Local Memory Window. With an Extended Send Ring the Tigon Window Base Register may have to be adjusted to allow this transfer.

   If Host based send rings are specified then this step is skipped and replaced by a DMA transfer started by the NIC following step 4.
3. The Host updates the Send Producer Index, and writes it into mailbox 2 in the Shared Memory region.
4. The NIC receives an internal mailbox event informing it that the Send Producer Index has been modified.
5. The NIC starts to DMA the frame from the Host to the transmit buffer in the NIC and enqueues the frame on the MAC Transmit ring. Once the DMA of the frame is complete, the Frame is transmitted onto the Network.
6. The NIC starts a DMA of the Send Consumer Index to notify the Host that the frame pointed to by the buffer descriptor(s) has been consumed. The DMA goes to the address specified in the Send Consumer Pointer . This DMA is subject to “Send BD Coalescing”, i.e. if coalescing is enabled, then the DMA may not be generated. The NIC may interrupt the host at this time.
7. If the Host was interrupted, or is already processing events or send and receive updates, it gets the new value of the send consumer and can now free the frame, as no further references are made to it. This step takes place only if the DMA was generated in step 6, above, and an Interrupt was generated in step 6 above or the host was already processing events or send and receive updates.

During send processing, the Host may be interrupted, although it is not mandatory. When the Host is only sending frames, interrupts are caused by meeting the send BD coalescing threshold or as soon as a send consumer index is DMA’ed (in the case where no send BD coalescing is used). Interrupts are also subject to Interrupt Avoidance (See “Interrupt Avoidance”, section 2.5.2)

2.9 Receive Flow Diagram

The following diagram shows how a single frame is received from the network into the NIC and into the Host. For this example, the frame fits into a single buffer descriptor.

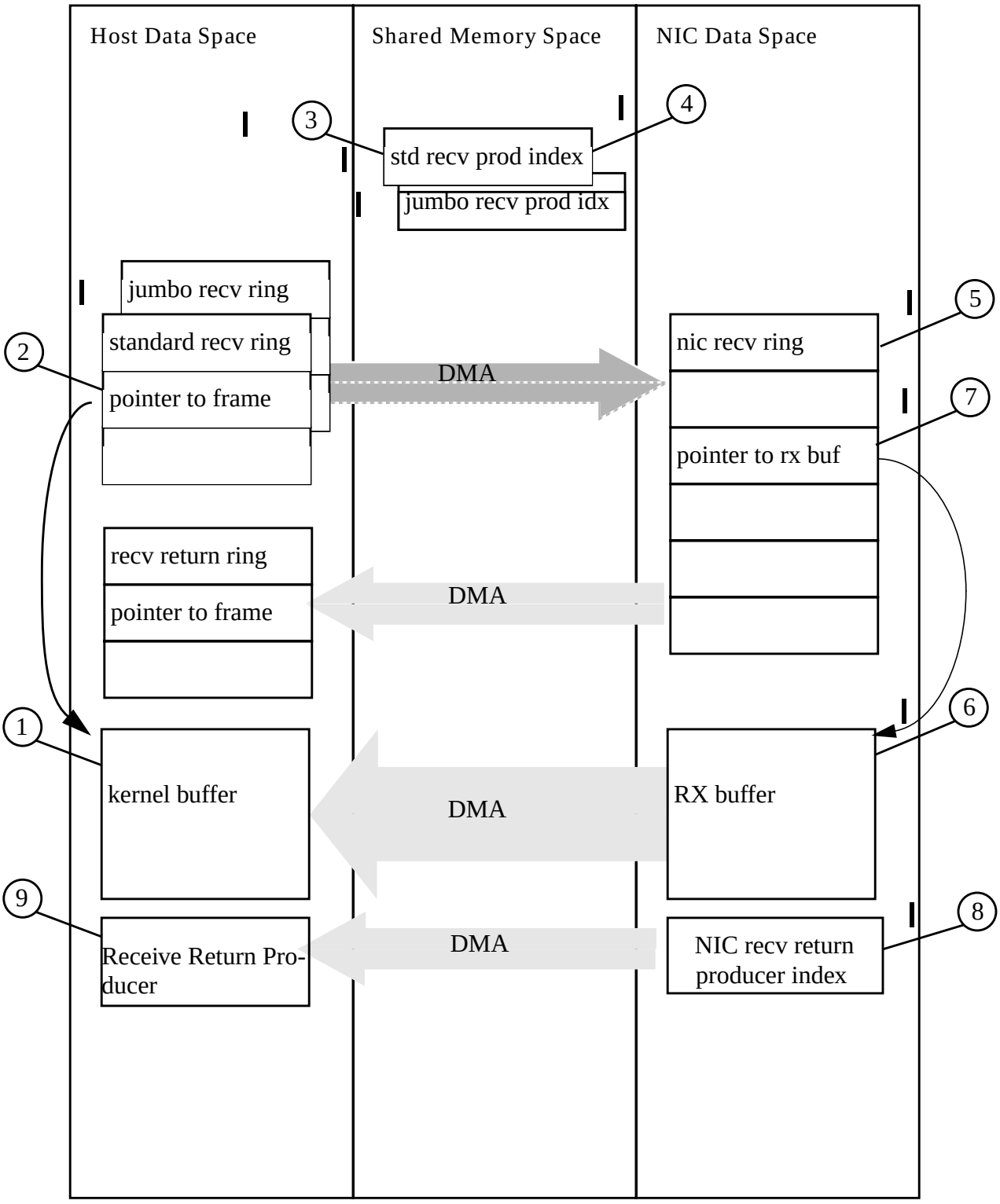


Figure 9. Receive Flow Diagram

1. The Host allocates a buffer in Host memory.
2. The Host creates a buffer descriptor that describes the buffer (standard or jumbo) and places it in the correct Host resident Receive Rings (Please See “Receive Rings”, section 2.4.2.3).
3. The Host updates the corresponding Ring Producer Index (“Mailboxes” on page 18).
4. The NIC receives an internal mailbox event informing him that a Receive Ring Producer Index has been modified and starts the DMA of the new buffer descriptor from the appropriate Receive Ring to the NIC copy of that Receive Ring.
5. The DMA of the new buffer descriptor completes and the NIC now waits for a frame to arrive from the network.
6. A frame arrives from the network. The NIC immediately begins to DMA the frame into the Host buffer described by the buffer descriptor. If the frame is larger than the value specified in the max_len field of the standard ring RCB’s max_len field the NIC will use a buffer from the Jumbo Ring if one is available. If a Jumbo buffer is not available or the packet is of less than max_len size the NIC will use one or more buffers described in the Standard Ring.
7. When the reception of the frame completes, the NIC updates the length and the flags fields in the buffer descriptor in the NIC copy of the Receive Ring and starts the DMA of the buffer descriptor(s) back into the Host Receive Return Ring (Please See “Receive Rings”, section 2.4.2.3).
8. The DMA of the filled receive buffer descriptor(s) completes. The NIC starts a DMA of the Receive Return Ring Producer Index to notify the Host that the frame pointed to by the buffer descriptor has been filled. The DMA goes to the address specified in the Receive Return Ring Producer Pointer. This DMA is subject to the “Receive BD Coalescing”, i.e. if the coalescing is enabled, this DMA may not be generated. The NIC may interrupt the host at this time.
9. If the Host was interrupted or was already processing events or send and receive updates, it gets the new value of the Receive Return Ring Producer Index and can now use the filled buffer, as no further references are made to it by the NIC. This step occurs only if a DMA was generated in step 11 above, and an Interrupt was generated in step 11 above, or the host was already processing events or send and receive updates.

During receive processing, the Host may be interrupted, although it is not mandatory. When the Host is only receiving frames, interrupts are caused by meeting the receive BD coalescing threshold or as soon as Receive Return Ring Producer Index is DMA’ed (when no receive BD coalescing is used).

Note that the old mechanism of placing a TG_CMD_SET_RECV_PRODUCER_INDEX or TG_CMD_SET_RECV_JUMBO_PRODUCER_INDEX is still acceptable, but the new direct mechanism is preferred.

2.10 Error Processing

The following classes of errors may be encountered:

- Error associated with a frame
- NIC internal error
- Host software invalid operation

When the NIC encounters an error when receiving a frame, the frame is discarded and the statistics are updated.

When the NIC encounters an error in sending a frame, the frame is discarded and statistics are updated.

When the NIC encounters invalid control information in Host memory or an internal inconsistent state the NIC halts its CPU. Host software is expected to reset the NIC.

2.11 Frame Filtering

The NIC does frame filtering based on the combination of settings of Promiscuous Mode (See "TG_CMD_SET_PROMISC_MODE", section 3.2.10), Multicast Mode (See "TG_CMD_SET_MULTICAST_MODE", section 3.2.14) and the specified local MAC address (See "MAC Address", section 2.1.1.4.1.)

If neither Promiscuous Mode nor Multicast Mode are specified the NIC will only accept and forward to the host unicast frames that are addressed directly to the MAC address specified by the host in the MAC address field of the General Communications area, as well as broadcast frames, and multicast frames that have their multicast addresses added to the multicast filtering list (See "TG_CMD_EXT_ADD_MULTICAST_ADDR", section 3.2.18 and See "TG_CMD_EXT_DEL_MULTICAST_ADDR", section 3.2.19.)

In Promiscuous Mode all non-errored frames received by the NIC are forwarded to the host.

In Multicast Mode all non-errored multicast frames are forwarded to the host as well as all broadcast and correctly addressed unicast frames.

3 Data Structures

3.1 Buffer Descriptors

Buffer descriptors are the elements of the Send and Recv rings. Individually, a buffer descriptor describes an area within Host memory where data is waiting to be transmitted from or received into. The largest amount of data that can be referenced by a single buffer descriptor is 64K -1 (16 bits in length). Multiple buffer descriptors may be chained together to provide scatter/gather for a single frame. The descriptors may point to any byte boundary. The only limit to the number of descriptors that describe a single Frame is the size of the Send or Receive Ring.

3.1.1 Receive Buffer Descriptors

Receive Buffer Descriptors (See figure 10 on page 57) are used for the reception of data from the network. There are 512 receive buffer descriptors in the Standard Receive Ring, 256 receive buffer descriptors in the Jumbo Receive Ring, 1024 buffer descriptors in the optional Mini Receive Ring, and either 1024 or 2048 buffer descriptors in the Receive Return Ring. See “Ring Control Blocks”, section 2.1.2.2.

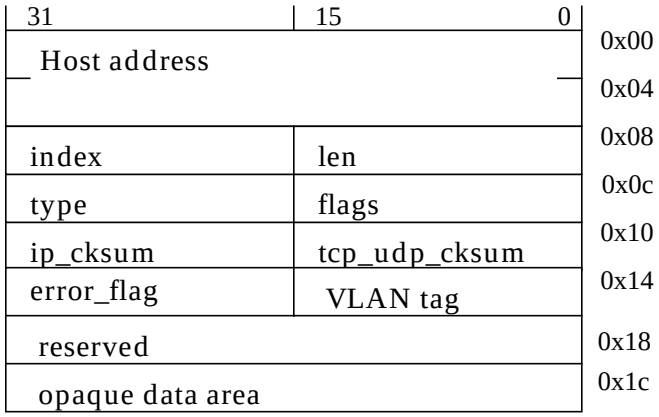


Figure 10. Receive Buffer Descriptor

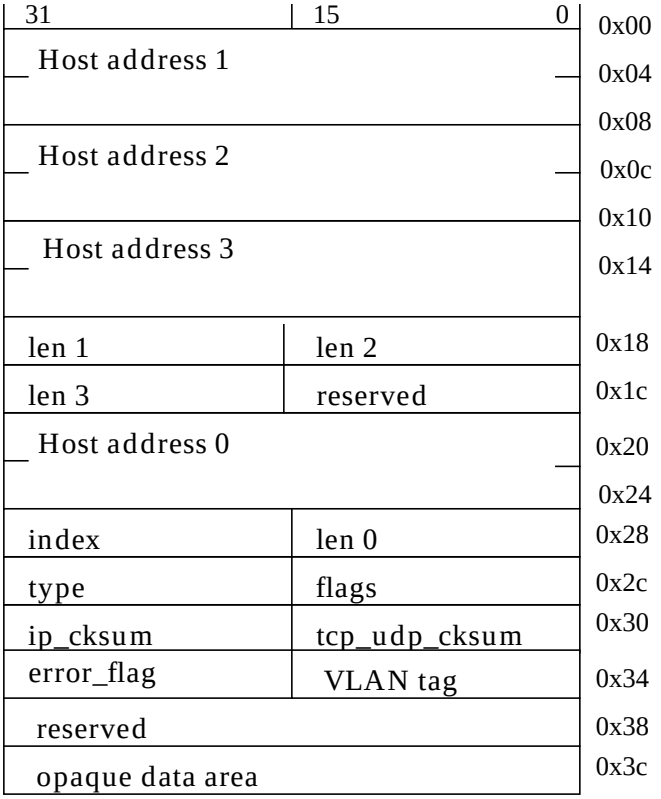


Figure 11. Extended Receive Buffer Descriptor

The Host address field contains the address of the buffer in Host memory. The Host address is in Host address format (See “Host Addresses”, section 1.3.1). For the Extended Receive Buffer Descriptor this address it the first address in the list of addresses.

The Host Address n field in the Extended Receive Buffer Descriptor contains the address of the nth piece of the buffer in Host memory.

The index field is used by the host to keep track of the position of the returned buffer in the Standard or Jumbo Receive Ring. The field is passed through opaquely by the NIC.

The len field is initially set by the Host and specifies the length of the buffer available for receiving data. On receive, the NIC modifies the length field to correspond with the amount of valid data in the buffer. A value of 0 indicates that there is no valid data in the buffer. For Extended Receive Buffer Descriptors this length field is the length of the data pointed to by Host address 0.

The len n field in the Extended Receive Buffer Descriptor contains the length of the nth piece of the buffer in Host memory.

The type field is used by the NIC internally and should be ignored and need not be set by the Host.

The Flag bits are used to indicate special processing that is needed in the buffer. Bits that are not explicitly defined here must be zero. See Table 26 for details.

The ip_cksum field is the checksum of the entire IP header. A correct checksum is 0 or 0xFFFF.

The tcp_udp_cksum field is the checksum of all data following the IP header, for the length defined in the IP header. If the RCB_NO_PSEUDO_HDR_CKSUM bit is set then the pseudo header checksum is not added to this value. If the bit is set this value includes the pseudo header.

The error_flag field contains a bitmask of possible errors. It is only valid if the BD_FLAGS_FRAME_HAS_ERROR bit is set in the flags field.

Table 25. Errored Frame Flags

Bit	Name	Description
0x00010000	BD_ERR_BAD_CRC	This frame has a bad CRC.
0x00020000	BD_ERR_COLL_DETECT	This frame had a collision.
0x00040000	BD_ERR_LINK_LOST_DURING_PKT	The link was lost while this, incomplete frame was being received.
0x00080000	BD_ERR_PHY_DECODE_ERR	The frame had an unspecified frame decoding error.
0x00100000	BD_ERR_ODD_NIBBLED_RCVD_MII	The packet arrived with an odd number of nibbles.
0x00200000	BD_ERR_MAC_ABORT	The MAC aborted the packet due to an unspecified internal error.
0x00400000	BD_ERR_LEN_LT_64	The MAC received a short packet.
0x00800000	BD_ERR_TRUNC_NO_RESOURCES	The MAC could not receive this entire packet due to a lack of internal resources.
0x01000000	BD_ERR_GIANT_FRAME_RCVD	This frame was longer than the maximum packet length value set in the ifMtu field (See “ifMtu”, section 2.1.1.4.15).

The VLAN tag field is filled in if the BD_FLAGS_VLAN_TAG bit is set in the flags field. It is the 2 byte VLAN tag that has been extracted from a 802.1q compliant frame.

The Reserved field is used internally by the Alteon NIC and does not have to be transferred between the NIC and the Host. The host should ignore the value of this field.

The opaque data field is reserved for the driver and any data placed here will be passed opaquely by the driver from the receive buffer descriptor in the standard or jumbo receive ring to the receive return ring.

3.1.2 Send Buffer Descriptors

Send Buffer Descriptors (Figure 12) are used for the transmission of data to the network. There are 128, 256 or 512 send buffer descriptors.

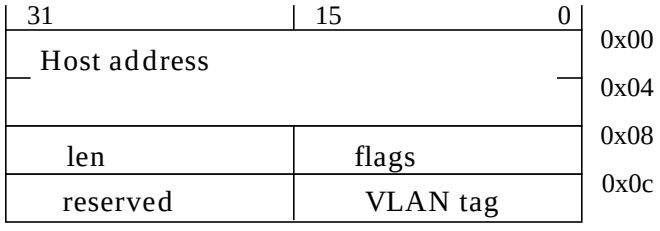


Figure 12. Send Buffer Descriptor

The Host address field contains the address of the buffer in Host memory. A length of 0 indicates that the descriptor does not have a buffer associated with it. The Host address is in Host address format (See “Host Addresses”, section 1.3.1).

The VLAN tag field is only used when the BD_FLAG_INSERT_VLAN_TAG is set. It should be set to the sixteen bit VLAN tag that should be inserted into the 802.1q compliant frame. For transmit this flag should only be set on the buffer descriptor that contains the Ethernet header.

The Flag bits are used to indicate special processing that is needed in the buffer. Bits that are not explicitly defined and detailed in Table 26 below must be set to zero. In order to generate the correct statistics one of the BD_FLAG_UCAST_PKT, BD_FLAG_MCAST_PKT, or BD_FLAG_BCAST_PKT bitfields should be set in the same buffer descriptor that has the BD_FLAG_END bit set. This is only necessary if you are not running with send checksumming enabled.

Table 26. Buffer Descriptor Flags

Bit	Name	Description
0x0001	BD_FLAG_TCP_UDP_CKSUM	Generate the TCP/UDP checksum for the frame that starts with this descriptor and ends with the descriptor with the BD_FLAG_FRAME_END bit set. Update the frame with the calculated checksum before transmitting it.
0x0002	BD_FLAG_IP_CKSUM	Generate the IP checksum for the IP header contained within this descriptor. It is assumed that the entire IP header is contained within a single descriptor.
0x0004	BD_FLAG_END	The frame ends at the end of the data in this buffer descriptor. This is set by the Host on transmit and by the NIC on recv.
0x0010	BD_FLAG_JUMBO_RING	Indicates that this packet came from the Jumbo Receive Ring, not the Standard Receive Ring (For Receive BDs only). This must be set by the driver, it is just copied through opaquely by the NIC firmware.
0x0020	BD_FLAG_UCAST_PKT	Unicast packet (part of 2 bit field).
0x0040	BD_FLAG_MCAST_PKT	Multicast packet that is not a broadcast packet (part of 2 bit field).
0x0060	BD_FLAG_BCAST_PKT	Broadcast packet (part of 2 bit field).
0x0080	BD_FLAG_IP_FRAG	This BD is part of an IP fragmented frame.
0x0100	BD_FLAG_IP_FRAG_END	Last BD in the last fragment in a train of fragments.
0x0200	BD_FLAG_VLAN_TAG	IThe frame associated with this buffer descriptor has an 802.1q VLAN tag associated with it.
0x0400	BD_FLAG_FRAME_HAS_ERROR	An error was detected by the NIC. The error detected is set in the error_flag word of the receive buffer descriptor.
0x0800	BD_FLAG_COAL_NOW	This bit is an explicit indication that the Send Consumer Index should be updated when the buffer associated with this buffer descriptor has been DMAed to the NIC. An interrupt may or may not be generated depending on the state of the RCB_FLAG_COAL_UPDATE_ONLY bit and the interrupt avoidance mechanism.
0x1000	BD_FLAG_MINI_RING	Indicates that this packet came from the Mini Receive Ring, not the Standard Receive Ring (For Receive BDs only). This must be set by the driver, it is just copied through opaquely by the NIC firmware.

3.2 Commands

A Command is a directive to the NIC.

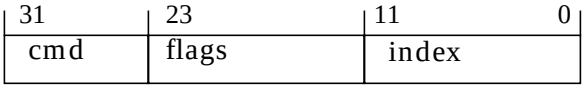


Figure 13. Command Definition

Cmd is a 1 byte field that specifies the Command.

Flags is a 12 bit field that gives additional command-specific information.

Index is a 12 bit field that contains the index that the command relates to, or what the index is to be set to, depending on the context of the command.

3.2.1 Extended Commands

There are some commands that are “extended”. That is, they take multiple command slots and the information following the actual command is somewhat free form. The details of the free form information is described in the individual commands. All such commands start with TG_CMD_EXT.

3.2.2 TG_CMD_HOST_STATE

Cmd: 0x01

Index: 0

Flags:

Table 27. TG_CMD_HOST_STATE flags

Value	Meaning
0x001	Host stack up
0x002	Host stack down

This command is sent by the Host to inform the NIC of the state of its stack. The Host state is initialized to DOWN. There is no acknowledgment to this command.

3.2.3 TG_CMD_FDR_FILTERING

Cmd: 0x02

Index: 0

Flags:

Table 28. TG_CMD_FDR_FILTERING flags

Value	Meaning
0x001	Enable software filtering
0x002	Disable software filtering

This command is sent by the Host to inform the NIC to use software filtering of MAC addresses instead of hardware filtering. This is used to overcome a bug in the original Tigon MAC that will cause the occasional loss of frames in environments where the NIC could receive a large quantity of frames not intended for it. This can happen when attached to a full duplex repeater.

This command is not implemented for Tigon 2 ASICs.

There is no acknowledgement to this command.

3.2.4 TG_CMD_SET_RECV_PRODUCER_INDEX (OBSOLETE)

Cmd: 0x03

Index: New recv producer index

Flags: 0

This command is sent by the Host to inform the NIC that there are Receive Buffer Descriptors ready to be consumed from the Standard Receive Ring. There is no acknowledgment to this command.

This command still works but has been obsoleted by a direct mechanism. (See Section “Mailboxes“).

3.2.5 TG_CMD_UPDATE_GENCOM_STATS

Cmd: 0x04

Index: 0

Flags: 0

This command is sent by the Host to inform the NIC that there is new data in the General Communications Area and it should be reread. Only the ifIndex and ifMTU fields are reread. This command is typically used to reset the ifMTU field to establish the threshold where packets should be marked as too long. There is no acknowledgment to this command.

3.2.6 TG_CMD_RESET_JUMBO_RING

Cmd: 0x05

Index: 0

Flags: 0

This command is sent by the Host to inform the NIC that it should stop using the Jumbo Receive Ring. It is typically used to retrieve the buffers associated with the Jumbo Receive Ring when no interface is using Jumbo Frames. This command is acknowledged by the TG_EVENT_RESET_JUMBO_RING.

3.2.7 TG_CMD_SET_PARTIAL_RECV_COUNT

Cmd: 0x06

Index: special

Flags: special

This command is sent by the Host to inform the NIC that it should not DMA the entire packet. Instead it should DMA the number of bytes specified in the concatenation of the index and the flags field. That is the Index and flags field are treated as a single 24 bit field. The maximum value allowed is 65532 (64k - 4). A value of 0 indicates that the entire frame should be DMAed. Note that this command does not effect the value placed into the received buffer descriptor length field.

This command only operates if receive checksum is disabled. If this command is issued and receive checksumming is enabled a TG_EVENT_ERROR event of type “Invalid command” is generated by the NIC and the command is ignored.

3.2.8 TG_CMD_ADD_MULTICAST_ADDR (OBSOLETE)

Cmd: 0x08

Index: 0

Flags: 0

This command is sent by the Host to inform the NIC of a new address in the Multicast MAC Address Transfer Buffer (See “Multicast MAC Address Transfer Buffer (OBSOLETE)”, section 2.1.1.4.3). The NIC updates its multicast filtering list by adding this new MAC address and acknowledges the command with a Multicast List Updated Event (See “TG_EVENT_MULTICAST_LIST_UPDATED”, section 3.3.5).

Although this mechanism is still accepted, this command has been superceded by the TG_CMD_EXT_ADD_MULTICAST_ADDR command.

3.2.9 TG_CMD_DEL_MULTICAST_ADDR (OBSOLETE)

Cmd: 0x09

Index: 0

Flags: 0

This command is sent by the Host to inform the NIC of a new address in the Multicast MAC Address Transfer Buffer (See “Multicast MAC Address Transfer Buffer (OBSOLETE)”, section 2.1.1.4.3). The NIC updates its multicast filtering list by deleting this MAC address and acknowledges the command with a Multicast List Updated Event. (See “TG_EVENT_MULTICAST_LIST_UPDATED”, section 3.3.5).

Although this mechanism is still accepted, this command has been superceded by the TG_CMD_EXT_DEL_MULTICAST_ADDR command.

3.2.10 TG_CMD_SET_PROMISC_MODE

Cmd: 0x0a

Index: 0

Flags:

Table 29. TG_CMD_SET_PROMISCUOUS_MODE flags

Value	Meaning
0x001	Enable promiscuous mode
0x002	Disable promiscuous mode

Upon receiving this command, the NIC enables or disables the promiscuous mode of its MAC. There is no acknowledgment to this command.

3.2.11 TG_CMD_LINK_NEGOTIATION

Cmd: 0x0b

Index: 0

Flags:

Table 30. TG_CMD_SET_LINK_NEGOTIATION flags

Value	Meaning
0x000	Renegotiate link on both PHYs
0x001	Renegotiate link on Gigabit PHY
0x002	Renegotiate link on 10/100 PHY

Upon receiving this command, the NIC renegotiates link negotiation based on the flags in the Gigabit and/or the 10/100 Link Negotiation field of the Tuning Parameters structure (See “Gigabit Link Negotiation”, section 2.1.1.4.10.7 and See “10/100 Link State”, section 2.1.1.4.18). There is no acknowledgement to this command.

3.2.12 TG_CMD_SET_MAC_ADDR

Cmd: 0x0c

Index: 0

Flags: 0

Upon receiving this command, the NIC rereads its MAC Address Register (See “MAC Address”, section 2.1.1.4.1) and updates the MAC portion of the NIC. Note that this does not affect outgoing frames as they are built by the Host. There is no acknowledgment to this command.

3.2.13 TG_CMD_CLEAR_PROFILE

Cmd: 0x0d

Index: 0

Flags: 0

Upon receiving this command, the NIC clears all of the profiling data. This command is available only if the driver has Profiling Enabled. Profiling is a compile time option and is disabled by default. Therefore, unless it is enabled at compile time, this command will result in an unknown command. There is no acknowledgment to this command.

3.2.14 TG_CMD_SET_MULTICAST_MODE

Cmd: 0x0e

Index: 0

Flags:

Table 31. TG_CMD_SET_MULTICAST_MODE flags

Value	Meaning
0x001	Enable Multicast mode
0x002	Disable Multicast mode

Upon receiving this command, the NIC enables or disables the multicast mode of its MAC. When multicast mode is enabled all multicast packets are delivered to the host, regardless of the setting of the multicast filtering list. When multicast mode is disabled then only those multicast packets that are in the multicast filtering list are delivered to the host. This command does not affect the current entries in the multicast filtering list. If further entries are added to or deleted from the multicast filtering list while the NIC is in multicast mode the updated state will be used when multicast mode is exited. There is no acknowledgment to this command.

3.2.15 TG_CMD_CLEAR_STATS

Cmd: 0x0f

Index: 0

Flags: 0

Upon receiving this command, the NIC clears all stats. There is no acknowledgment to this command.

3.2.16 TG_CMD_SET_RECV_JUMBO_PRODUCER_INDEX (OBSOLETE)

Cmd: 0x10

Index: New recv producer index

Flags: 0

This command is sent by the Host to inform the NIC that there are Receive Buffer Descriptors ready to be consumed from the Jumbo Receive Ring. There is no acknowledgment to this command.

This command still works but has been obsoleted by a direct mechanism. (See Section “Mailboxes“).

3.2.17 TG_CMD_REFRESH_STATS

Cmd: 0x11

Index: 0

Flags: 0

This command is sent by the Host to inform the NIC that it should update its current statistics. The new statistics are found in the structure pointed to by the Refresh Stats Pointer. There is a possibility that if the NIC is receiving or sending packets during the execution of this command some statistics could be slightly incorrect. It is therefore recommended that this command only be executed when the NIC is configured to not receive packets and while no packets are being sent. There is no acknowledgment to this command.

3.2.18 TG_CMD_EXT_ADD_MULTICAST_ADDR

Cmd: 0x12

Index: 0

Flags: 0

Extension: 2 commands

This command is sent by the Host to inform the NIC of a new Multicast address that the NIC should accept. The address follows the command as a command extension in the next two command slots. The format of the standard MAC address format (See “MAC Addresses”, section 1.3.3). The NIC updates its multicast filtering list by adding this new MAC address and acknowledges the command with a Multicast List Updated Event (See “TG_EVENT_MULTICAST_LIST_UPDATED”, section 3.3.5).

3.2.19 TG_CMD_EXT_DEL_MULTICAST_ADDR

Cmd: 0x13

Index: 0

Flags: 0

Extension: 2 commands

This command is sent by the Host to inform the NIC of a Multicast address that the NIC should no longer accept. The address follows the command as a command extension in the next two command slots. The format of the standard MAC address format (See “MAC Addresses”, section 1.3.3). The NIC updates its multicast filtering list by deleting this MAC address and acknowledges the command with a Multicast List Updated Event (See “TG_EVENT_MULTICAST_LIST_UPDATED”, section 3.3.5).

3.3 Events

An Event is a signal to the Host that something has occurred that the Host should be aware of. Generally speaking, all Events cause interrupts. However, the events can be coalesced and the interrupts can be avoided. Please See “Interrupt Avoidance”, section 2.5.2 for details.

Events are 8bytes long.

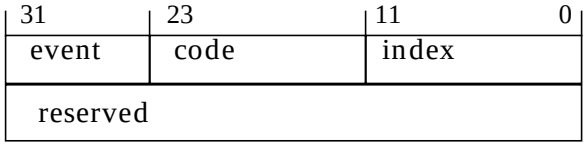


Figure 14. Event Definition

Event is a one byte field that specifies the Event.

Code is a 12 bit field that gives event-specific information.

Index is a 12 bit field that gives additional code specific information.

3.3.1 TG_EVENT_NIC_FIRMWARE_OPERATIONAL

Event: 0x01

Code: 0

Index: 0

This event is the first event that the NIC generates. The Host will not issue any commands to the NIC until this event has been received. The Host is interrupted when this event is generated.

3.3.2 TG_EVENT_STATS_UPDATED

Event: 0x04

Code: 0

Index: 0

This event is generated whenever the NIC updates the statistics area in Host memory. The Host may also use this event to reset its watchdog timer. It is recommended that the TG_EVENT_STATS_UPDATED event be used as a watchdog event. Should the NIC not generate this event within a reasonable period of time (See “Stat Ticks”, section 2.1.1.4.10.3) you can assume that the NIC has died and appropriate action should be taken. The Host is interrupted when this event is generated (subject to the Interrupt Avoidance mechanism described in Section 2.5.2 “Interrupt Avoidance“).

3.3.3 TG_EVENT_LINK_STATE_CHANGED

Event: 0x06

Code:

Table 32. TG_EVENT_LINK_STATE_CHANGED codes

Value	Meaning
0x001	Gigabit link is now UP
0x002	Link is now DOWN
0x003	10/100 link is now UP

Index: 0

This event is generated whenever the link changes state. When the NIC is initialized, the carrier’s state is initialized to OFF. The host is interrupted when this event is generated (subject to the Interrupt Avoidance mechanism described in Section 2.5.2 “Interrupt Avoidance“).

3.3.4 TG_EVENT_ERROR

Event: 0x07.

Code:

Table 33. ERROR codes

Value	Meaning
0x001	Invalid command
0x002	Unimplemented command

Table 33. ERROR codes

Value	Meaning
0x003	Invalid configuration

Index: Command in question.

This event is generated whenever an error occurs. The host is interrupted when this event is generated (subject to the Interrupt Avoidance mechanism described in Section 2.5.2 “Interrupt Avoidance“).

3.3.5 TG_EVENT_MULTICAST_LIST_UPDATED

Event: 0x08.

Code:

Table 34. TG_EVENT_MULTICAST_LIST_UPDATED codes

Value	Meaning
0x001	Multicast address added
0x002	Multicast address deleted

Index: 0.

This event is generated in response to the Add Multicast Address (See “TG_CMD_ADD_MULTICAST_ADDR (OBSOLETE)”, section 3.2.8) or Delete Multicast Address (See “TG_CMD_DEL_MULTICAST_ADDR (OBSOLETE)”, section 3.2.9) command after reading the Multicast MAC Address Transfer Buffer (See “Multicast MAC Address Transfer Buffer (OBSOLETE)”, section 2.1.1.4.3) and updating the NIC’s multicast list appropriately. The host is interrupted when this event is generated (subject to the Interrupt Avoidance mechanism described in Section 2.5.2 “Interrupt Avoidance“).

3.3.6 TG_EVENT_RESET_JUMBO_RING

Event: 0x09

Code: 0

Index: 0

This event is generated in response to a TG_CMD_RESET_JUMBO_RING command. It is generated as soon as the consumer and producer of the jumbo ring have been disabled. The host is interrupted when this event is generated (subject to the Interrupt Avoidance mechanism described in Section 2.5.2 “Interrupt Avoidance“)

4 PERFORMANCE TUNING

Performance tuning is an art. The following are a few pointers to key areas that you should first study and use to exerpiment. Good luck!

4.1 Reducing the Host/NIC interaction

This API is designed to reduce the number of times that information must be passed between the Host and the NIC. This means that the amount of information passed on any interaction must be large. It also means that whenever possible the simplest and fastest mechanism should be used to carry the information. This reduction of interaction is done through coallescing work for multiple packets into single information packets and by reducing interrupts through “Interrupt Avoidance”.

4.1.1 Information Coallescing

The NIC Firmware is designed to pass information to the Host interrupt the Host in chunks. The advantage of provide information in chunks is that it reduces the number of times the Host and NIC must interact. The disadvantage is that it can create latencies. This is a trade off that must be carefully tested by the Host driver designer. Setting the coalescing thresholds too high can cause longer latencies that you desire. The NIC firmware does attempt to reduce these latencies, however by not coallescing in cases where it has been a “long time” since the last bit of information has been passed.

The mechanism used for receive packet coalescing is the Recv BD Coalescing parameter found in the Tuning Parameters area. This parameter controls the number of packets that NIC will receive before updating the Receive Return Ring Producer Index. The NIC updates the Receive Return Ring Producer Index immediately if a hard clock tick has occured since the last update (100 microseconds). After the Recv Return Ring Producer Index has been updated there may be an interrupt generated subject to the Interrupt Avoidance algorithms.

If the Recv BD Coalescing counter is set to zero, the Receive Return Ring Producer Index will be updated every time a frame is received generating extra DMA activity.

The Send BD Coalescing count reduces the number of times the Send Consumer Index is updated. For example, if this count is set to 60, one update of the Send Consumer Index will be done for every 60 frames transmitted. The number of clock ticks that must expire before the Send Consumer Index is updated regardless of the current coalescing count is set using the Send Coalesced Ticks paramter. The buffer descriptor also has an explicit return flag (See “Buffer Descriptor Flags”, section Table 26.)

If the Send BD Coalesing counter is set to zero, the Send Consumer Index will be updated every time a frame has been transmitted.

The developer is encouraged to experiment with these values in order to obtain a good balance on your particular operating system.

4.1.2 Interrupt Avoidance

In addition to the counts described above, the NIC firmware guarantees that it will not generate an interrupt if it finds that the Host is already in the Interrupt Handler or processing Events. The NIC firmware uses two mechanisms to determine if the Host is already in the Interrupt Handler or processing events.

First, the NIC firmware inspects the Interrupt State Bit in the Miscellaneous Host Control register. (See Section 2.1.1.2.1 “Tigon Miscellaneous Host Control Register.) The Interrupt State Bit, when set, indicates that the Host is already processing interrupts or events and the NIC firmware simply enqueues the new event or updates the appropriate index without generating an interrupt. This mechanism is for platforms where the Host processes all the events while in staying in the Interrupt Handler.

Second, the NIC firmware inspects Mailbox 0 (See Section 2.1.1.3 “Mailboxes). If a 1 is written in Mailbox 0, the NIC firmware assumes that the Host is already processing interrupts or events and will not generate an Interrupt. This mechanism is for Hosts which do not perform all work in the Interrupt Handler, but rather dispatch a task (or thread of execution) to process events and index updates.

For the second type of Host, the first act of the task or thread is to write a 1 in mailbox 0 and store the value of the event, send, and receive indices. Just prior to writing a 0 to mailbox 0 and exiting the task or thread, the value of the indices must be compared to the previously stored values to determine if there is more work to do. A change in oen of the values indicates the arrival of additional work which must be handled, since the NIC will not generate an interrupt for them because there is a 1 in Mailbox 0.

Finally, for send interrupts the interrupt mechanism can be further defeated by setting the RCB_FLAG_COAL_UPDATE_ONLY flag in the Send RCB. This bit might be used if your driver usually wishes to reap send completion events from the Send Packet handler.

4.2 Receive Ring

The NIC firmware is designed to use multiple Receive Buffer Descriptors at a time. Developers are encouraged to send empty Receive Buffer Descriptors to the NIC in blocks, rather than one at a time, to reduce the per-frame overhead.

4.3 Send Ring

To reduce per-frame overhead, it is recommended that the developers initialize all the fields in the Send Ring Descriptors once and update only those fields that are necessary for each frame. For example, if the same buffers are reused, the address field does not need to be rewritten over and over.

4.4 Transmitting Latencies and Buffer Descriptors

The current version of the firmware takes approximately 5 microseconds to process a buffer descriptor (from the point where it reads the buffer descriptor until a DMA is begun on an idle NIC). In order to get the best performance the host driver should coalesce small buffer descriptors into a single buffer descriptor until the DMA time of the operation is approximately 5 microseconds. For example, on a host system that has read DMA performance of 100 MB/s the host should coalesce small buffers (when possible) until 500 bytes have been coalesced.

4.5 NIC Data Buffer Sizes

The NIC's internal data buffering can be controlled by the Transmit Buffer Ratio parameter of the General Communications Area. By changing this ratio you may find performance differences on systems that are limited in bus performance in either the read or the write direction. The use of Jumbo Frames may also affect this ratio. We have found that unless Jumbo Frames are being used 64 KB of transmit buffer is plenty. However for Jumbo enabled drivers you may improve performance by increasing this number to as much as 256 KB (about a 5/16 ratio for a 1 MB card).

4.6 PCI Command Memory Write and Invalidate

One some hosts there is a significant difference in performance between the PCI command Memory Write and the PCI command Memory Write and Invalidate. The Memory Write and Invalidate command can be made to work on both Tigon and Tigon 2 ASICs.

4.6.1 Memory Write and Invalidate on the Tigon ASIC

To use the Memory Write and Invalidate command on the original Tigon ASIC you must set it up so that it is *always* used (that is, Memory Write commands cannot be generated). Setting this up is very simple, just set the PCI Write Command field of the PCI State Register to 0xF instead of the standard 7 (See "Tigon PCI State Register", section 2.1.1.2.4). This will cause the entire cache line to be invalidated every time a write DMA occurs, so it is important to ensure that any place where data is going to be written using a DMA can handle a full cache line invalidation.

The sample driver code is set up to allow this to happen, so long as the cache line is 32 bytes long or smaller. In particular, the send, recieve and event indices are placed in their own 32 byte cache lines. Also the Receive Buffer Descriptor is 32 bytes long and the array is aligned on a 32 byte boundary to ensure that writing any descriptor will not cause a problem with any other previously written descriptor. Finally, the data receive buffers are allocated in a way to ensure that cache lines will not overlap other data.

4.6.2 Memory Write and Invalidate on the Tigon 2 ASIC

While somewhat harder to set up, the Tigon 2 ASIC will use the Memory Write and Invalidate in a way that is transparent to cache line boundaries. That is, when it is possible to use this command it will, but if the entire cache line is not being written it will not use this command. If your host has a bus implementation where Memory Write is significantly slower than Memory Write and Invalidate you may want to implement the cache line tricks described above. (See "Memory Write and Invalidate on the Tigon ASIC", section 4.6.1)

To set up the use of this command make sure that you set the PCI_CONF_CMD_MEM_WRITE_INVALIDATE bit in the PCI Configuration Command Register (See PCI Local Bus Specification Revision 2.1 [1]). Also ensure that your BIOS has properly set up the Cache Line Length field of the PCI Configuration region. Finally, set the Write Max DMA field of the PCI State Register to indicate the same size as your the Cache Line Length field of the PCI Configuration region. (See "Tigon PCI State Register", section 2.1.1.2.4)

4.7 PCI Command Memory Read Multiple

The NIC has two modes of operation for choosing which PCI Read Memory command to use. These modes are selected using the “Use Memory_Read_Multiple PCI Command” bit of the Tigon PCI State Register (See “Tigon PCI State Register”, section 2.1.1.2.4). The following table describes when each command is used. Normal Command Set refers to when the “Use Memory_Read_Multiple PCI Command” bit is not set. The Optional Command Set refers to when it is set.

Table 35. Memory Read Command Usage

Number of Words	Normal Command Set	Optional Command Set
< 1/2 the cache size or 1-2 words if cache size disabled	Memory Read	Memory Read
> 1/2 the cache size or 3-15 words if cache size disabled	Memory Read Line	Memory Read Multiple
> cache size or > 16 words if cache size disabled	Memory Read Multiple	Memory Read Multiple

4.8 PCI Burst Length

For best performance, experiment with the Read Max DMA and Write Max DMA fields in the PCI State Register (detailed in Section 2.1.1.2.4). The default setting of 0 was selected for the sample driver because it provides optimal performance for most systems. Note that the use of the Memory Write and Invalidate command on Tigon 2 ASICs requires that the value for the Write Max DMA field be set to the Cache Line Length.

4.9 Checksum Offload

Checksum offload can significantly descrease the amount of time that the Host’s CPU spends processing a packet. Many systems spend as much as 15% of their time just calculating checksums. In general, if the host can support checksum offload there will be a major performance gain. However, if you cannot support checksum offload in your host be sure you do not set any of the flags in the RCBs that would enable these features. Enabling these features requires different algorithms in the NIC that can adversely effect performance.

4.10 DMA Read Errata on Tigon 2 ASICs

4.10.1 Problem Description

When the Tigon 2 (Rev. 6) ASIC performs a read transaction, it can expect several responses ranging from completing the cycle to being told to retry it. In the event of a retry, the device must re-issue the same command to the same address in order to be compliant. Occasionally in certain operating modes, the Tigon 2 ASIC does not reissue the exact same command.

4.10.2 Details

The ASIC may issue a Memory Read Multiple (MRM) with either a Memory Read Line (MRL) or a Memory Read (MR) command issued during the retry. It might also try a MRL with a MR on the retry. Either one of these conditions violates the PCI spec.

The Tigon ASIC uses the number of words remaining in the DMA to determine what read command to issue. From the time the length register changes until the correct command can be driven on the PCI bus is three cycles. Unfortunately, under certain conditions, the ASIC will begin a new transaction in two cycles. In this situation, the Tigon can send out the old read command. This isn't a problem by itself, although it may be inefficient for the memory sub-system. However, if this access is retried, the length register will be fully updated and a different read command may come out.

These three conditions must be met for the problem to occur.

- The Tigon ASIC must begin a read transaction within two cycles of finishing a previous read transaction.
- The number of words remaining after the previous transaction must cross a full or half cacheline boundary.
- The command must be retried.

Your bridge chip may or may not create the third condition. If it does then you must by properly configuring the Tigon ASIC to avoid either the first or the second condition. If your bridge chip does not cause a command to be retried then you need not worry about this errata.

4.10.3 Work-around Options

The first condition occurs only when the Tigon is voluntarily stopping DMA on programmable power of boundaries. In this configuration, the Tigon ends all DMA bursts when the boundary is crossed. It does not stop requesting the bus, so if the bus grant is still active, it will begin another transaction until the grant is removed or the DMA is finished. Any condition that forces the ASIC to drop its bus request will guarantee that it cannot begin another access within 2 cycles of the end of a previous one. This behavior is controlled by the Read_Max_DMA bits in the PCI state register. When these bits are set to zero (Disabled), the Tigon will keep the bus until told to stop or until the DMA is finished. It will not be able to generate accesses that occur two cycles apart. (See "Tigon PCI State Register", section 2.1.1.2.4).

The second condition can be worked around by forcing the read transactions to ask for more data. When the Tigon ASIC realizes it doesn't need more than a cacheline of data, it changes the read command from MRM to something smaller (MRL or just MR.) If the ASIC is forced to use the larger commands all the time, than the retry command will always agree with the original command. To accomplish this, the use Mem_Read_Multiple bit must be set and the default PCI read command must be a MRM. These bits are also located in the PCI state register. (See "Tigon PCI State Register", section 2.1.1.2.4).

4.10.4 Impact Observations

Each workaround has potential performance impacts. The first workaround could change the bus utilization of the Tigon ASIC. If the bus grant is inactive when the ASIC ends a burst, the TIGON has lost control of the bus until the grant returns. If the Max DMA bits are zero, the ASIC will not give up the bus until the latency timer expires. If the latency timer is longer than the Max DMA time, changing the Max DMA time to unlimited could result in the Tigon holding the bus longer because it will not release the bus until the latency timer expires. If the latency timer is shorter than the maxDMA time, disabling the max DMA time could also result in the ASIC giving up the bus less often. This analysis is extremely dependent upon the host system arbiter performance.

The second workaround will not change the bus utilization, but it might use the memory subsystem poorly. With this workaround, the Tigon could ask for a MRM even if it only needed a couple of bytes. Depending upon its design, the memory subsystem could have pre-fetched data that it doesn't need.

5 Firmware Initialization

This section describes firmware start-up and processing from power on, system reboot and Run Code restart. The bootstrap loader is kept in ROM. NIC PCI register values and various other manufacturing information such as the MAC Address are kept in the EEPROM. In addition, there are two firmware code objects in the EEPROM. They are:

- PCI Configuration and Memory Initialization
- Diagnostics.

NOTE: *At present, only Cable Integrity Check is supported. Other items will be supported in a future release.*

5.1 Power-on Bootstrap Sequence

The bootstrap sequence is:

1. Run ROM bootstrap loader and load step 2
2. Run PCI configuration and memory initialization code and load step 3
3. Run Diagnostics code and enter cable integrity loop

The ROM bootstrap loader checksums the bootstrap area of EEPROM and performs the 32 bit CRC check on the PCI configuration and memory initialization code as it loads the firmware code into the SRAM. Then it clears the ROM fail bit in the CPU_STATE register and executes the firmware.

The PCI configuration and memory initialization firmware initializes the PCI configuration registers and calls the bootstrap loader to load the diagnostics. The reason for this two stage process is that the PCI configuration registers must be set up before the host PCI BIOS detects the card.

The diagnostics code tests the address and data lines and SRAM integrity, flashes the LEDs, and then go into an operational mode which monitors the link state and lights the link LED whenever the link is established.

5.2 Hard Reset

A hard reset is due to system power up, PCI reset, or the Hard Reset bit in the Miscellaneous Host Control Register (See “Tigon Miscellaneous Host Control Register”, section Table 6.) The hard reset causes the bootstrap sequence to be executed.

5.3 Operation at System Reboot

When the Host system boots, the host driver executes its initialization. Part of this initialization is loading the Run Time Code into the NIC’s SRAM and starting it.

Host software reads the MAC address from the EEPROM during driver initialization and sets the MAC address register.

5.4 Firmware Download

In order to use the NIC you must download the runtime firmware provided with your open driver kit. See Section 5.5.1 “Runtime Program File. To download this code you must fill in the Text, Data, and Read-only segments with the data present in the appropriate array. This data is placed into NIC’s memory by copying it from the appropriate array into the NIC’s memory via the shared memory window. See Section 2.1.1.2.5 “Tigon Window Base Address Register. The BSS and short BSS segments must be zeroed.

NOTE: *Be certain to stop the NIC processor before attempting to download the firmware using the Halt Bit of the CPU State Register (See “Tigon CPU State Register”, section 2.1.1.2.8).*

Once the firmware is loaded it is prudent to verify that it is correct. Just compare the memory in the NIC against the original arrays. Again this must be done by using the shared memory window.

Finally, once the firmware is downloaded and verified set the Program Counter to the start address specified in the firmware header file and start the firmware using the NIC’s CPU State register (See “Tigon CPU State Register”, section 2.1.1.2.8). For Tigon 2 systems, CPU B must not be started by the Host, the initialization code in CPU A’s firmware starts CPU B.

The intialization code at the beginning of the firmware load has a set of instructions that will help the new driver devloper determine if the NIC has been downloaded correctly with respect to byte swapping and word swapping. The following table indicates failure address, CPU state register value and cause. The addresses specified assumes that the start address is 0x4000.

Table 36. Firmware Failure Locations

Failure Address	CPU State Register Value	Cause
0x4002	0x80000	Byte swapped only
0x4004	0x10000	Word swapped only
0x4003	0x80000	Byte and word swapped

5.5 Firmware Reload Operation

The host software loads the firmware in two cases:

1. The NIC halts. In this case the Host takes a firmware core dump, resets the NIC, runs host diagnostics, reloads the run code, re-initializes all data structures, and restarts the NIC code.
2. An administrative request is received. In this case the software halts the NIC, reloads the run code, re-initializes all data structures, and restarts the NIC code.

Once the run code is started, the firmware initializes the NIC. When initialization is complete the NIC informs the Host via a NIC_FIRMWARE_OPERATIONAL event.

5.5.1 Runtime Program File

The firmware file is a binary file in MIPS ELF format that has been converted into a C header file with the instructions and data placed into initialized arrays. The name of this file is alt_fw.h for the original Tigon ASIC and alt_fw2.h for the Tigon 2 ASIC. One or both should be included into your driver and the correct one downloaded depending on the revision of the ASIC. There are five different sections of the runtime firmware, Text, Data, Read-only data, BSS, and short BSS. In the alt_fw.h file the starting address and length of each segment are specified. For the Text, Data, and Read-only data segments the actual data associated with these segments is given in the appropriate initialized array.

5.6 Link Ready Operation

The NIC passes frames only when the optical link is established. The state of the link is indicated on the “Link” LED on the NIC bulkhead. On means that the Link is Up. Changes in link state are passed to the Host software using the LINK_STATE_CHANGED event.

After the NIC_FIRMWARE_OPERATIONAL event is posted, the Run Code posts a LINK_STATE_CHANGED event.

The NIC EEPROM code does not have link negotiation code in it. Link negotiation is accomplished via the NIC firmware.

6 Bibliography

The following documents are useful in understanding the context in which the NIC operates. It is always encouraged to use the latest revision of each document.

[1] PCI Special Interest Group, PCI Local Bus Specification Revision 2.1, 1995, June.

[2] IEEE, Overview and Architecture, ANSI/IEEE 802-1990.

[3] IEEE, Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specification, ANSI/IEEE Standard 802.3-1990.

[4] Postel, J., Internet Protocol, RFC 791, 1981 September; 45 p.

[5] Postel, J., User Datagram Protocol, RFC 768, 1980 August 28; 3 p.

[6] Postel, J., Transmission Control Protocol, RFC 793, 1981 September; 85 p.

[7] Kastenholz, F., Definitions of Managed Objects for the Ethernet-like Interface Types, RFC 1643, 1994 July; 19 p.

[8] Rose, M; McCloghrie, K.,eds., Management Information Base for network management of TCP/IP-based internets:MIB-II, RFC 1213, 1991 March; 70 p.

[9] McCloghrie, K.;et.al., eds., Evolution of the Interfaces Group of MIB-II, RFC 1573, January 1994