

DSS NETWORKS

GigMAC PMC & PCI Cards

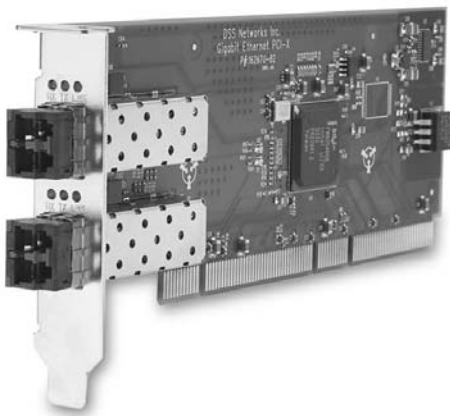
GigPMC Switch

Board and Driver Users Manual

(Includes all Intel-based models 5164, 5261, 5262, 5468 and 6267)
(copper and fiber models)

Document Version 2.5, P/N 131903

May 2006



1. INTRODUCTION.....	4
1.1 SEE ALSO.....	4
1.2 COMPATIBILITY	4
2. MODEL NUMBERS.....	5
3. KEY FEATURES.....	5
3. SWITCH CAPABILITIES (MODEL 5468 only).....	6
4. OEM DEVELOPER KIT CONTENTS.....	7
5. SYSTEM REQUIREMENTS	8
6. HARDWARE INFORMATION	9
6.1 BOARD PHOTOS	9
6.2 BOARD LED INDICATORS	9
6.2 BOARD LED INDICATORS	10
6.3 PMC CONNECTOR PIN/SIGNAL DEFINITIONS.....	12
6.4 PCI CONFIGURATION REGISTERS	16
6.4.1 VENDOR AND DEVICE IDS.....	17
6.5 EEPROM LISTINGS	17
6.6 DEFAULT REGISTER SETTINGS	19
6.7 ETHERNET FRAME LATENCY.....	21
7. POWER CONSUMPTION SPECS	21
8. HARDWARE INSTALLATION.....	21
8.1 INSTALLATION IN PC COMPUTER	22
8.2 EMBEDDED OR COMPACT PCI INSTALLATION.....	22
9. COPPER CABLING AND CONNECTOR INFO.....	23
9.1. FIBER CABLE SPECIFICATIONS.....	23
9.2 COPPER RJ-45 CONNECTOR AND CABLE.....	24
10. SOFTWARE DRIVER INSTALLATION.....	25
10.1 LINUX DRIVER INSTALLATION AND USAGE	25
10.2 VXWORKS DRIVER INSTALLATION AND USAGE	32
10.3 DRIVER UTILITY COMMAND REFERENCE	33
10.3.1 STATISTICS SHOW FUNCTION.....	33
10.3.2 PCI REGISTERS SHOW FUNCTION.....	34
10.3.3 MAC REGISTERS SHOW FUNCTION.....	35
10.3.4 MAC STATISTICS SHOW FUNCTION.....	36
10.3.5 PHY REGISTERS SHOW FUNCTION	38
10.3.6 EEPROM SHOW FUNCTION.....	39
10.3.7 MAC ADDRESS SHOW FUNCTION	40
10.3.7 MAC ADDRESS SHOW FUNCTION	40
10.3.8 BUFFER DESCRIPTOR STATUS SHOW FUNCTION.....	40
10.3.9 STARTING FRAME GENERATOR.....	41
10.3.10 STOPPING FRAME GENERATOR	41
10.3.11 SWITCH STATUS SHOW FUNCTION.....	42
10.4 DPM DRIVER MANAGEMENT API.....	42
10.5 SWITCH CONFIGURATION / MANAGEMENT API	44
11. TESTING AND VERIFICATION	46
12. SPECIFICATIONS	47
12.1 ENVIRONMENTAL SPECIFICATIONS	50
13. WARRANTEE AND SUPPORT INFO.....	51

1. INTRODUCTION

The GigMAC and GigPMC-switch family of network adapter cards and switches are a high-performance, cost-effective solution for adding Gigabit Ethernet connectivity to any embedded or real-time network appliance or network access device equipped with a PCI slot or PMC mezzanine site and is ideal for embedded systems utilizing add-on modules including PCI, Compact PCI and VME based systems.

The GigMac and GigPMC-switch family includes optimized, high-performance driver support for VxWorks and Linux. Instructions for Linux driver installation are included in this manual. In addition, a companion document titled “VxWorks Users Manual and Integration Guide” is provided for technical assistance in integrating and testing the driver in an embedded real-time VxWorks environment.

1.1 SEE ALSO

Please also see the following documents on our website at www.dssnetworks.com and also included in the OEM developers kit CD:

Datasheets – please see product datasheets and other updated product information on OEM developer CD and on website.

Release Notes -- where updated information is provided on new features, compatibility, performance benchmarks, platform information and corrected problems.

VxWorks Users Manual and Integration Guide, DSS Document part no 131901. Provides technical information on integrating and testing our controllers and drivers into a VxWorks BSP and system board.

GIGFAQ.HTML – Also on website and included on OEM developer CD contains many answers to commonly asked questions regarding Gigabit Ethernet and our products including performance and system recommendations.

README.LINUX – Included on OEM developer CD contains latest driver installation and usage instructions for Linux Operating System.

netPerformance.txt -- Included on OEM developer CD contains useful information on tuning the VxWorks network protocol stack.

1.2 COMPATIBILITY

The GigMAC and GigPMC-Switch family is fully compliant with the following standards:

- IEEE 802.3 (all sections applicable to 1000 Base T, 1000 Base SX, 1000 Base LX)
- IEEE 802.1D and IEEE 802.1Q as applicable for VLAN and priority queuing support
- PCI 2.2 and PCI-X 1.0 compliant
- PCI low-profile specification (as applicable for model)
- IEEE 1386.1 Draft 2.2
- **Linux driver compatibility:** Standard Linux 2.4 or 2.6 kernel level network driver module compiled and tested on Intel and PowerPC architectures up to Linux kernel 2.4.26 and 2.6.4.
- **VxWorks driver compatibility:** Standard Tornado 2.0.2 and Tornado 2.2/2.2.1, VxWorks 5.4/5.5/5.5.1 Enhanced Network Driver loadable module integrated and tested on PowerPC and Intel architectures

2. MODEL NUMBERS

This user manual covers all Models of our GigMAC and GigPMC-switch PMC and PCI cards including:

PMC models

- 5164 quad-port (copper)
- 5261-LC (fiber)
- 5261-RJ (copper)
- 5262-LC (fiber)
- 5262-RJ (copper)
- 5468 (GigPMC-Switch, copper)

PCI-X models

- Model 6267-SFP (fiber and copper, PCI-X)

3. KEY FEATURES

The GigMAC and GigPMC-switch offer the following key features:

- Sustained throughput of 245 Mbytes/sec (1.96 Gb) per port over PCI bus using 64-bit, 66 MHZ PCI
- Sustained throughput of 118 Mbytes/sec (944 Gb) per port over PCI bus using 32-bit, 33 MHZ PCI
- Frame processing rate of up to 1,000,000 frames per second (as measured on 2 GHZ P4/Zeon running Linux 2.4 kernel)
- Very low latency, < 2 microseconds for short frames
- Support for copper and fiber interfaces
- Driver support for vxWorks and Linux including embedded Linux
- Built in management and diagnostics capabilities in drivers
- High-performance frame generator (wire and bus-speed capable)
- Performance and health monitoring statistics
- 64-bit PMC and 64-bit PCI/PCI-X low-profile card adapter form factors
- Supports 33 and 66 MHz, 32 and 64 bit PCI interfaces including PMC form factor
- Support for 133/100/66 PCI-X in Models 5262, 5468 and 6267-SFP
- Installs in any PCI or CompactPCI system with a PMC or universal PCI slot
- Ideal solution for CompactPCI and embedded 1U and 2U PC-based systems
- Utilizes either 5 or 3.3 volt pci bus power supply and contains onboard power regulators
- Full duplex Gigabit Ethernet Interface over standard CAT5e cabling
- Complies with all PCI revision 2.2 mechanical and electrical requirements
- Fully IEEE 802.3z, IEEE 802.3ab, 802.3u and IEEE 1386 compliant
- Compatible with all 10/100/1000BaseT hubs, switches and routers
- Burst rate of up to 256 dwords (1024 bytes) over PCI bus
- Jumbo frame support for up to 9K, 802.3x full duplex flow control with automatic pause and priority with multiple priority queues

3. SWITCH CAPABILITIES (MODEL 5468 only)

The model 5468 provides an onboard Gigabit Ethernet layer 2 switch with management capabilities. The operating registers in the switch are accessible via a register-level API that is provided in the host driver and described in section 10.5 of this manual.

The switch can be programmed to support advanced features including following:

- ARL table control (4K MAC-address management)
- VLAN management (4K VLANS)
- Trunking and failover control
- Port mirroring control
- QoS / 802.1P / Priority / Differential Services
- Port traffic control
- Switch status
- PHY (transceiver) control
- Jumbo frame control
- Port rate control

4. OEM DEVELOPER KIT CONTENTS

An OEM developer kit is provided with the purchase of the Gigabit Ethernet controllers which contains drivers, documentation and sample code including the following:

- Driver source code for vxWorks, Linux 2.4, PowerPC, Intel platforms
- Users Manual(s)
- VxWorks Users Manual and Integration Guide
- Datasheets for chipset controllers (Intel, National, etc.)
- TCP/UDP/IP performance test programs (vxWorks, Linux, Windows versions included)
- TCP, UDP and raw driver performance tests
- Driver Utilities (Linux)
- High-performance frame generator (wire and bus-speed capable)
- Transmit and receive callbacks (hooks) for driver-level application code
- Internal and external loopback capabilities
- Built-in performance instrumentation statistics
- Gigabit Ethernet FAQ sheet
- NetPerformance.txt protocol stack tuning guide for vxWorks

5. SYSTEM REQUIREMENTS

Intel Platform (minimum): Pentium III 800 MHZ or faster with PC100 or PC133 SDRAM, **32-bit 33 MHZ PCI**.

Intel Platform (recommended): Pentium4 or Xeon at 1.8GHZ or faster, DDR SDRAM or RDRAM based, **64-bit, 66 MHZ PCI or PCI-X (133/100/66 MHZ)**.

PowerPC Platform (minimum): PowerPC 7400 or 750 series at 500 MHZ or faster, PC100 or PC133 SDRAM, 32-bit, 33 MHZ PCI

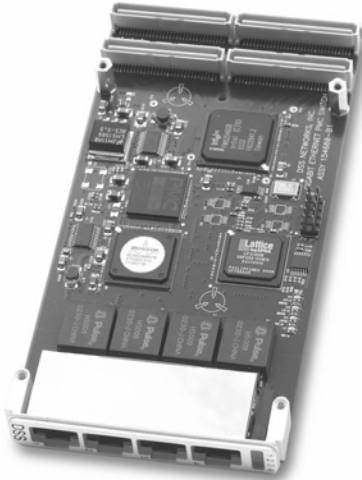
PowerPC Platform (recommended): PowerPC 7400 or 750 series at 700 MHZ or faster, PC133 SDRAM or DDR SDRAM, 64-bit, 33 or 66 MHZ PCI

Linux Operating system: Linux 2.4 (2.4.18 - 2.4.26), Linux 2.6 (all)

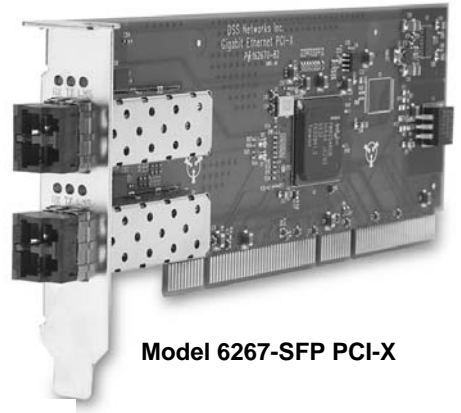
Tornado II requirements: Tornado 2.2/vxWorks 5.5 (or Tornado 2.0.2/vxWorks 5.4 with patch updates. PCI Configuration Library support (pciConfigLib).

6. HARDWARE INFORMATION

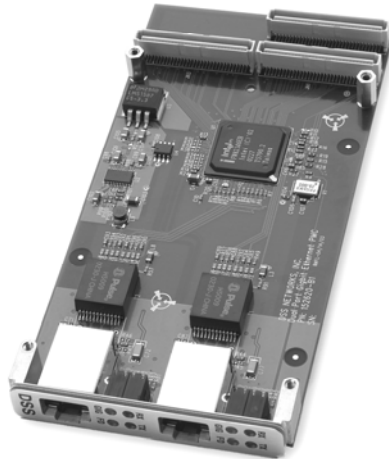
6.1 BOARD PHOTOS



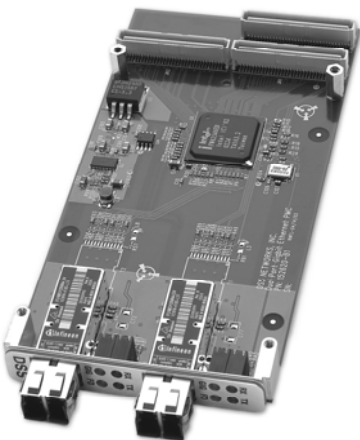
Model 5468 GigPMC 8-port switch



Model 6267-SFP PCI-X



Model 5262-RJ PMC



Model 5262-LC fiber PMC



Model 5164 - 4-port PMC

6.2 BOARD LED INDICATORS

PCI / PMC Boards LED Function Table

Gigabit NIC Model	LED # 1 Green	LED # 2 Green	LED #3 Yellow	LED #4 Yellow
PCI-X 6267	TX	RX	LINK	N/A
PMC 5468	LNK/ACT	LNK/ACT	LNK/ACT	LNK/ACT
PMC 5164	LNK/ACT	LNK/ACT	LNK/ACT	LNK/ACT
PMC 5262	TX	RX	LINK	FD

Model 6267-SFP

The front panel has onboard pluggable SFP fiber optic connectors that support fiber optic cabling using 50 or 62.5 micron multimode fiber with LC type connectors. An RJ-45 type 1000 Base T SFP connector is also available for this model. There are 3 LED indicators per port that provide the following as marked with silkscreen nomenclature on front panels:

- ❑ LED_LINK (link)
- ❑ LED_TX (transmit)
- ❑ LED_RX (receive)

Link “LNK” Indicator

On solid green to indicate auto-negotiation complete and MAC and driver level link is up.

Transmit and Receive Activity Indicators

Transmit and Receive LEDs are solid or blinking yellow to indicate activity on the link (blinking shows tx and rx activity).

Model 5262-RJ dual-port or 5261-RJ single port PMC (copper)

The front panel has two onboard RJ-45 connectors that supports the connection of Category 5e cabling with 4 LED indicators per port that provide the following as marked with silkscreen nomenclature on front panels:

- ❑ LED_1000 (1000 Mb mod)
- ❑ LED_FD (full-duplex mode)
- ❑ LED_TX (transmit activity)
- ❑ LED_RX (receive activity)

Model 5262-LC dual-port or 5261-LC single port PMC (fiber)

The front panel has two onboard fiber LC connectors that support the connection of singlemode or multimode fiber optics with 4 LED indicators per port that provide the following as marked with silkscreen nomenclature on front panels:

- ❑ LED_1000 (1000 Mb mod)
- ❑ LED_FD (full-duplex mode)
- ❑ LED_TX (transmit activity)
- ❑ LED_RX (receive activity)

Model 5164 quad-port PMC with RJ-45 connector(s)

The front panel has four onboard RJ-45 connectors that support the connection of Category 5e cabling with 1 LED indicator per port that provides both a “link” and “activity” indicator as follows:

- ❑ OFF (no link)
- ❑ ON (link)
- ❑ BLINKING (link and activity)

Model 5468 GigPMC Switch with 4-port RJ-45 connector(s)

The front panel has four onboard RJ-45 connectors that support the connection of Category 5e cabling with 1 LED indicator per port that provides both a “link” and “activity” indicator as follows:

- ❑ OFF (no link)
- ❑ ON (link)
- ❑ BLINKING (link and activity)

6.3 PMC CONNECTOR PIN/SIGNAL DEFINITIONS

PMC Connectors Pin Assignments

Pn1/Jn1 32 Bit PCI

1		TCK	-12V	NC	2
3		GND	INITA#		4
5		INTB#	INTC#	NC	6
7		BUSMODE1#	+5V		8
9	NC	INTD#	PCI-RSVD	NC	10
11		GND	PCI-RSVD	NC	12
13		CLK	GND		14
15		GND	GNT#		16
17		REQ#	+5V		18
19		VIO	AD[31]		20
21		AD[28]	AD[27]		22
23		AD[25]	GND		24
25		GND	C/BE[3]#		26
27		AD[22]	AD[21]		28
29		AD[19]	+5V		30
31	NC	VIO	AD[17]		32
33		FRAME#	GND		34
35		GND	IRDY#		36
37		DEVSEL#	+5V		38
39		GND/PCIXCAP	LOCK#		40
41	NC	SDONE#	SBO#		42
43		PAR	GND		44
45	NC	VIO	AD[15]		46
47		AD[12]	AD[11]		48
49		AD[09]	+5V		50
51		GND	C/BE[0]#		52
53		AD[06]	AD[05]		54
55		AD[04]	GND		56
57	NC	VIO	AD[03]		58
59		AD[02]	AD[01]		60
61		AD[00]	+5V		62
63		GND	REQ64#		64

Pn2/Jn2 32 Bit PCI

1	NC	+12V	TRST#		2
3		TMS	TDO		4
5		TDI	GND		6
7		GND	PCI-RSVD	NC	8
9	NC	PCI-RSVD	PCI-RSVD	NC	10
11	NC	BUSMODE2#	+3.3V		12
13		RST#	BUSMODE3#	PMC_1	14
15		3.3V	BUSMODE4#	NC	16
17	NC	PCI-RSVD	GND		18
19		AD[30]	AD[29]		20
21		GND	AD[26]		22
23		AD[24]	3.3V		24
25		IDSEL	AD[23]		26
27		+3.3V	AD[20]		28
29		AD[18]	GND		30
31		AD[16]	C/BE[2]#		32
33		GND	PMC-RSVD	NC	34
35		TRDY#	+3.3V		36
37		GND	STOP#		38
39		PERR#	GND		40
41		+3.3V	SERR#		42
43		C/BE[1]#	GND		44
45		AD[14]	AD[13]		46
47		M66EN	AD[10]		48
49		AD[08]	3.3V		50
51		AD[07]	PMC-RSVD	NC	52
53		+3.3V	PMC-RSVD	NC	54
55	NC	PMC-RSVD	GND		56
57	NC	PMC-RSVD	PMC-RSVD	NC	58
59		GND	PMC-RSVD	NC	60
61		ACK64#	+3.3V		62
63		GND	PMC-RSVD	NC	64

Pn3/Jn3 32 Bit PCI

1	NC	PCI-RSVD	GND		2
3		GND	C/BE[7]#		4
5		C/BE[6]#	C/BE[5]#		6
7		C/BE[4]#	GND		8
9	NC	VIO	PAR64		10
11		AD[63]	AD[62]		12
13		AD[61]	GND		14
15		GND	AD[60]		16
17		AD[59]	AD[58]		18
19		AD[57]	GND		20
21	NC	VIO	AD[56]		22
23		AD[55]	AD[54]		24
25		AD[53]	GND		26
27		GND	AD[52]		28
29		AD[51]	AD[50]		30
31		AD[49]	GND		32
33		GND	AD[48]		34
35		AD[47]	AD[46]		36
37		AD[45]	GND		38
39	NC	VIO	AD[44]		40
41		AD[43]	AD[42]		42
43		AD[41]	GND		44
45		GND	AD[40]		46
47		AD[39]	AD[38]		48
49		AD[37]	GND		50
51		GND	AD[36]		52
53		AD[35]	AD[34]		54
55		AD[33]	GND		56
57	NC	VIO	AD[32]		58
59	NC	PCI-RSVD	PCI-RSVD		60
61	NC	PCI-RSVD	GND		62
63		GND	PCI-RSVD	NC	64

Pn4/Jn4 (optional I/O – Model 5468 only)

1	NC	I/O	I/O	NC	2
3	NC	I/O	I/O	NC	4
5	NC	I/O	I/O	NC	6
7	NC	I/O	I/O	NC	8
9	NC	I/O	I/O	NC	10
11	NC	I/O	I/O	NC	12
13	NC	I/O	I/O	NC	14
15	NC	I/O	I/O	NC	16
17	NC	I/O	I/O	NC	18
19	NC	I/O	I/O	NC	20
21	NC	I/O	I/O	NC	22
23	NC	I/O	I/O	NC	24
25	NC	I/O	I/O	NC	26
27	NC	I/O	I/O	NC	28
29	NC	I/O	I/O	NC	30
31	NC	I/O	I/O	NC	32
33	NC	I/O	I/O	NC	34
35	NC	I/O	I/O	NC	36
37	NC	I/O	I/O	NC	38
39	NC	I/O	I/O	NC	40
41	NC	I/O	I/O	NC	42
43	NC	I/O	I/O	NC	44
45	NC	I/O	I/O	NC	46
47	NC	I/O	I/O	NC	48
49	NC	I/O	BYP/GND		50
51		BYP/GND	SGOUT-P7		52
53		SGIN-P7	SGOUT-N7		54
55		SGIN-N7	BYP/GND		56
57		BYP/GND	SGOUT-P8		58
59		SGIN-P8	SGOUT-N8		60
61		SGIN-N8	BYP/GND		62
63		BYP/GND	I/O	NC	64

6.4 PCI CONFIGURATION REGISTERS

in: unit: 0, bus: 16, dev: 1, func: 0, pci regs:

```
device&vendor: 10108086
  stsAndErrs: 02300017
  membaseCsr: 00020004
  flashbaseCsr: 00000000
  iobaseCsr: 00001001
  membaseUpper: 00000000
    irq: 00ff0106
  cfgLat/cache: 00802408
```

6.4.1 VENDOR AND DEVICE IDS

All Intel based models use the 82546GB chip which has the following vendor and device ID:

Vendor ID: 8086
Device ID: 1010

6.5 EEPROM LISTINGS

Model 5468

```
in: unit: 0, i82546 serial eeprom cfg:
  eeprom[0x00]: 5000
  eeprom[0x01]: 0cc2
  eeprom[0x02]: 2e99
  eeprom[0x03]: 0420
  eeprom[0x08]: a921
  eeprom[0x09]: 1104
  eeprom[0x0a]: 460b (INIT_CTRL_1)
  eeprom[0x0b]: 5468
  eeprom[0x0c]: dbad
  eeprom[0x0d]: 1010
  eeprom[0x0e]: 8086
  eeprom[0x0f]: 3c68 (INIT_CTRL_2)
  eeprom[0x10]: 00c3
  eeprom[0x11]: 1010
  eeprom[0x12]: 0000
  eeprom[0x13]: 2100
  eeprom[0x14]: 1ac8 (INIT_CTRL_3-B, IntrPin: 1, TBI/Serdes: 1)
  eeprom[0x20]: 00c3
  eeprom[0x21]: 7861
  eeprom[0x22]: 280c
  eeprom[0x23]: 2100
  eeprom[0x24]: 0ac8 (INIT_CTRL_3-A, IntrPin: 0, TBI/Serdes: 1)
  eeprom[0x2f]: 0609 (LED2_MODE: 6, LED0_MODE: 9)
  eeprom[0x30]: 002c
  eeprom[0x31]: 4000
  eeprom[0x32]: 1107
  eeprom[0x33]: 0000
  eeprom[0x34]: 002c
  eeprom[0x35]: 4000
  eeprom[0x3f]: 7d23
```

Model 5262-RJ

in: unit: 0, i82546 serial eeprom cfg:

```
eeprom[0x00]: 5000
eeprom[0x01]: 0cc2
eeprom[0x02]: 1e99
eeprom[0x03]: 0420
eeprom[0x08]: a921
eeprom[0x09]: 1104
eeprom[0x0a]: 460b (INIT_CTRL_1)
eeprom[0x0b]: 5262
eeprom[0x0c]: dbad
eeprom[0x0d]: 1010
eeprom[0x0e]: 8086
eeprom[0x0f]: 3468 (INIT_CTRL_2)
eeprom[0x10]: 010d
eeprom[0x11]: 1010
eeprom[0x12]: 0000
eeprom[0x13]: 2100
eeprom[0x14]: 18c8 (INIT_CTRL_3-B, IntrPin: 1, TBI/Serdes: 0)
eeprom[0x20]: c30d
eeprom[0x21]: 7861
eeprom[0x22]: 280c
eeprom[0x23]: 2100
eeprom[0x24]: 08c8 (INIT_CTRL_3-A, IntrPin: 0, TBI/Serdes: 0)
eeprom[0x2f]: 0609 (LED2_MODE: 6, LED0_MODE: 9)
eeprom[0x30]: 002c
eeprom[0x31]: 4000
eeprom[0x32]: 1107
eeprom[0x33]: 0000
eeprom[0x34]: 002c
eeprom[0x35]: 4000
eeprom[0x3f]: d895
```

6.6 DEFAULT REGISTER SETTINGS

The following lists the default register settings for the Intel 82546 Gigabit controller on the cards (copper models):

in: unit: 0, ioAddr: 40020000, mac regs:

```
CTRL: (0x0000):      0x18f41ae1 (PRST: 0, TFCE: 1, RFCE: 1, RST: 0, LRST: 0,
FD: 1)
                                (FRCD: 1, FRCS: 1, SPD: 2, ILOS: 1, ASDE: 1)
STATUS: (0x0008):    0x0000db83 (PCIX: 0, BU64: 1, PC66: 1, TBI: 0, TXOF: 0)
                                ( LU: 1, FD: 1)
EECD: (0x0010):      0x110
EERD: (0x0014):      0xd8953f10
EXT-CTRL: (0x0018):  0xc00
MDIC: (0x0020):      0x14204140
FCAL: (0x0028):      0xc28001
FCAH: (0x002c):      0x100
FCT: (0x0030):       0x8808
VET: (0x0038):       0x8100
ICR: (0x00c0):       0x0
ITR: (0x00c4):       0x0
ICS: (0x00c8):       0x0
IMS: (0x00d0):       0x1d0df
IMC: (0x00d8):       0x1d0df
RCTL: (0x0100):      0x643822a (EN: 1, BAM: 1, bsize: 3)
FCTTV: (0x0170):     0x200
TXCW: (0x0178):     0x1a0
RXCW: (0x0180):     0xc000000
TCTL: (0x0400):     0x204000a
TIPG: (0x0410):     0x802008
LEDCTL: (0x0e00):    0x7060f09
PBA: (0x1000):       0x100030
RDBAL: (0x2800):     0x817792b0
RDBAH: (0x2804):     0x0
RDLEN: (0x2808):     0x1000
RDH: (0x2810):       0x0
RDT: (0x2818):       0xff
RDTR: (0x2820):     0x32
RADV: (0x282c):     0x32
TDBAL: (0x3800):     0x817782b0
TDBAH: (0x3804):     0x0
TDLEN: (0x3808):     0x1000
TDH: (0x3810):       0x0
TDT: (0x3818):       0x0
TIDV: (0x3820):     0x32
TXDCTL: (0x3828):   0x1040202
TADV: (0x382c):     0x32
RAL[0]: (0x5400):    0xcc25000
RAH[0]: (0x5404):    0x80001e99
RAL[1]: (0x5408):    0xcc25000
RAH[1]: (0x540c):    0x8000fe90
RAL[2]: (0x5410):    0xcc25000
RAH[2]: (0x5414):    0x8000ff90
```

The following lists the default register settings for the Intel 82546 Gigabit controller on the cards (fiber models):

```
in: unit: 0, ioAddr: e087c000, mac regs:
  CTRL: (0x0000):      0x18f40280 (PRST: 0, TFCE: 1, RFCE: 1, RST: 0, LRST: 0,
FD: 0
                                (FRCD: 0, FRCS: 0, SPD: 2, ILOS: 1, ASDE: 0)
  STATUS: (0x0008):    0x00007ba3 (PCIX: 1, BU64: 1, PC66: 1, TBI: 1, TXOF: 0)
                                ( LU: 1,  FD: 1)
  EECD: (0x0010):      0x110
  EERD: (0x0014):      0x92970210
EXT-CTRL: (0x0018):    0x800c00
  MDIC: (0x0020):      0x14290200
  FCAL: (0x0028):      0xc28001
  FCAH: (0x002c):      0x100
  FCT: (0x0030):       0x8808
  VET: (0x0038):       0x8100
  ICR: (0x00c0):       0x0
  ITR: (0x00c4):       0x0
  ICS: (0x00c8):       0x0
  IMS: (0x00d0):       0x1d0df
  IMC: (0x00d8):       0x1d0df
  RCTL: (0x0100):      0x440823a (EN: 1, BAM: 1, bsize: 0)
  FCTTV: (0x0170):     0x200
  TXCW: (0x0178):      0x800001e0
  RXCW: (0x0180):      0xcc0041e0
  TCTL: (0x0400):      0x204000a
  TIPG: (0x0410):      0x802008
  LEDCTL: (0x0e00):    0x7060f09
  PBA: (0x1000):       0x100030
  RDBAL: (0x2800):     0x3f5000
  RDBAH: (0x2804):     0x0
  RDLEN: (0x2808):     0x1000
  RDH: (0x2810):       0x0
  RDT: (0x2818):       0xff
  RDTR: (0x2820):      0x32
  RADV: (0x282c):      0x32
  TDBAL: (0x3800):     0x3f4000
  TDBAH: (0x3804):     0x0
  TDLEN: (0x3808):     0x1000
  TDH: (0x3810):       0x0
  TDT: (0x3818):       0x0
  TIDV: (0x3820):      0x32
  TXDCTL: (0x3828):    0x1040202
  TADV: (0x382c):      0x32
  RAL[0]: (0x5400):    0xcc25000
  RAH[0]: (0x5404):    0x80009297
  RAL[1]: (0x5408):    0xcc25000
  RAH[1]: (0x540c):    0x8000fe90
  RAL[2]: (0x5410):    0xcc25000
  RAH[2]: (0x5414):    0x8000ff90
```

6.7 ETHERNET FRAME LATENCY

The following table shows typical system level frame latencies measured during high-performance testing using frame generator including path from SDRAM to wire through system controller across PCI bus interface:

Frame size (bytes)	Typical latency (microseconds)
60 – 100	< 2
500	4
1000	8
1500	12

7. POWER CONSUMPTION SPECS

PCI-X & PMC Boards Power Consumption Table

Card Model	3.3V Source Current I (mA, A)	Power (Watts)
PMC 5261	900	2.97
PMC 5262	1.17	3.86
PCI-X 6267	1.17	3.86
PMC 5164	2.34	7.90
PMC 5468	1.93	6.40

8. HARDWARE INSTALLATION

Before attempting to install the card into your system, please make sure of the following:

Shut off the power to the computer and any peripherals. It is important to **remove the power cable** to the computer to reduce the possibility of residual power remaining in the power supply.

Ground yourself. Many electronic components inside computer and on the card can be severely damaged by receiving a shock of static electricity. Before touching any electronic components or boards, discharge any static electricity on your body by touching the bare metal case around the power supply inside your computer. Avoid excessive movement during the installation, such as walking across carpets, as this can generate static. If you must leave the installation area before the installation is complete, be sure to ground yourself again before continuing the installation.

Assess system power requirements. If you already have other PCI cards in your Mac, make sure that your system is able to provide the necessary power to support the addition of the adapter card. Check your computer's user manual for power limitations.

8.1 INSTALLATION IN PC COMPUTER

There are many different styles and types of PC platforms that utilize PCI slots. This section contains a generic installation procedure. Please refer to your User's manual for more detailed instructions on installing the adapter in a PC.

The PMC cards require a PMC-to-PCI Carrier adapter module in order to be used in a standard PCI system. Please refer to section 2.2 below for instructions on installing a PMC adapter onto a mainboard carrier adapter.

Step 1 - Shut down the power to the computer system and remove the power cord and any peripheral cables.

Step 2 - Open the computer case. This will expose the interior of the case and allow the adapter card to be installed. Consult your User's Manual for assistance in opening your computer. The PCI card edge connector contains "key" notches that correspond to dividers in the motherboard slot. It is very important to align the key notches with the matching dividers in the slot. Failure to align the notches correctly could cause the card or slot to be damaged.

Step 3 - Align the PCI edge connector and the system motherboard socket. Once aligned, push down firmly on the PCI card until it is completely seated in the slot. Once the adapter card has been successfully installed, close the PC case, re-attach the power cable and any other cables that were removed for the hardware installation procedure. Consult Section 2.3 for information on connecting the adapter to the network.

8.2 EMBEDDED OR COMPACT PCI INSTALLATION

There are many different styles and types of Embedded System platforms that utilize PMC Mezzanine slots. This section contains a generic installation procedure. Please refer to your User's manual for more detailed instructions on installing the adapter in an Embedded or CompactPCI system.

Step 1 - Shut down the power to the computer system and remove the power cord and any peripheral cables.

Step 2 – Remove the CompactPCI or Embedded Mainboard controller from the system, if necessary to access the PMC slot. This will allow the PMC adapter card to be more

easily installed. Consult your User's Manual for assistance in opening and removing the mainboard. The PMC card has 3 connectors labeled JN1, JN2 and JN3. JN1 and JN2 are required for all modes of operation while JN3 is required only for 64-bit mode. Your CompactPCI mainboard or other PMC carrier will also contain PMC card connectors labeled JN1 and JN2, and optionally JN3 and JN4. JN4 is for optional user-defined I/O and is only used by the Model 5468 GigPMC switch. Additionally, the mainboard may also contain either a 5 volt or 3.3 volt key to prevent voltage mismatches. Since the PMC card may operate from a 5 or 3.3 volt supply, it can be used in either a 5 or 3.3 volt keyed system. It is very important to align the connectors before pressing the PMC card onto the mainboard to prevent damage to the card or slot.

Step 3 - Align the PMC connectors and the system motherboard Connectors and push down firmly on the PMC card until it is completely seated in the connectors on the mainboard. Once the adapter card has been successfully installed, re-install the mainboard. Re-attach the power cable and any other cables that were removed for the hardware installation procedure.

9. COPPER CABLING AND CONNECTOR INFO

9.1. FIBER CABLE SPECIFICATIONS

Distance (Model 6267, 5261-LC, 5262-LC multimode/singlemode fiber)

(1000-base-SX 850nm multimode)
(1000-base-LX 1310nm singlemode)

1000BASE-SX/LX (850 nm Laser for multimode-SX, 1310nm laser for single-mode-LX)			
Fiber Core Diameter	Type	Fiber Bandwidth Mhz* km	Distance
62.5/125 um	multi-mode	160 Mhz * km	2 to 220 m
62.5/125 um	multi-mode	200 Mhz * km	2 to 275 m
50.0/125 um	multi-mode	400 Mhz * km	2 to 500 m
50.0/125 um	multi-mode	500 Mhz * km	2 to 550 m
8.0/125 um	single-mode	500 Mhz * km	5-10 km

Connecting Fiber optic Model 6267, 5261-LC, 5262-LC

This section explains how to connect the fiber cards to the external network using standard fiber optic cables. Typically 50 or 62.5 micron multimode fiber optic cables with

LC or SC type connectors are used depending on the connector option. For extended distance, single-mode fiber can be used in models equipped with extended range single-mode connectors.

Insert the fiber optic cable into the SC or LC type connector until the self-locking tab clicks into position. Connect the opposite end in to a 1000 Base SX switch. Two types of cables are used when connecting the fiber cards to the network. A workstation or "straight through" cable is typically used to connect Ethernet adapters to switches. A fiber "crossover" cable may also be used to connect controllers back-to-back. This configuration is useful for loopback and/or diagnostic purposes or when a switch is not available.

9.2 COPPER RJ-45 CONNECTOR AND CABLE

Connecting Copper RJ-45 Models 5261-RJ, 5262-RJ, 5164, 5468 & 6267-SFP (copper model)

This section explains how to connect the adapter cards to the external network using the standard Category 5e, 5e or 6 cables. The maximum cable length is typically 100 meters or 328 feet.

Insert the Category 5e cable into the RJ-45 connector until the self-locking tab clicks into position. Connect the opposite end in to a 10/100 or 10/100/1000 Base T switch. Two types of cables are used when connecting the adapter cards to the network. A workstation or "straight through" cable is typically used to connect Ethernet adapters to switches. A "crossover" cable may also be used to connect controllers back-to-back. This configuration is useful for diagnostic purposes or when a hub or switch is not available. However, it is not recommended for extended use, as it violates the IEEE specification for 10 Mbit, 100 Mbit and 1000 Mbit Ethernet networks.

Note(1): Models 5164, 5261-RJ, 5262-RJ, 5468 and 6267-RJ support "auto-MDIX" mode where a crossover cable is not required when directly attaching two controllers.

Note(2): Cables used for Gigabit networks must use all 8 wires. In 10 and 100 modes, wires are dedicated for transmit or receive while in Gigabit mode, data is transmitted and received over all 4-pairs.

RJ-45 pinouts for CAT5 connectors and cables are shown in the following table:

Pin	10/100 Signal	Gigabit Signal
1	Transmit+	Channel A+
2	Transmit-	Channel A-
3	Receive+	Channel B+
4	Unused	Channel C+
5	Unused	Channel C-
6	Receive-	Channel B-
7	Unused	Channel D+
8	Unused	Channel D+

10. SOFTWARE DRIVER INSTALLATION

The following sections explain how to install the driver software in VxWorks, and Linux based systems.

10.1 LINUX DRIVER INSTALLATION AND USAGE

These instructions assume that you are running Red Hat Linux 7.x, 8.x, 9.x or a similar Linux OS. You should be using at least a kernel version 2.4.18 installation for any other Red Hat-based system will be similar. For other Linux based systems, the location of some files may be slightly different.

Building the driver

To build the driver, copy the driver files to an appropriate directory. You will need to make sure your CD is mounted.

```
$ mkdir dpm
$ cd dpm
$ cp /mnt/cdrom/linux-dpm-driver/dpm/dpm*.tar.gz .
$ gunzip dpm*.tar.gz
$ tar vxf dpm*.tar
```

To create a new driver object module:

```
$ make clean
$ make
```

Repeat same procedure with driver utility directory as follows:

```
$ mkdir util
$ cd util
$ cp /mnt/cdrom/linux-dpm-driver/util/util*.tar.gz .
$ gunzip util*.tar.gz
$ tar vxf util*.tar
```

To create a new utility programs:

```
$ make clean
$ make
```

Note(1): If you are installing the adapter cards in an SMP machine, you should comment out the standard CFLAGS line in the “Makefile” and uncomment the SMP version of CFLAGS before compiling the driver (safe to leave SMP enabled on newer versions of Linux (2.4.18 – 2.4.26, 2.6).

Note(2): Before compiling, please also edit the Makefile and set “INCLUDEDIR” to the path of your Linux kernel source tree. For example:

```
INCLUDEDIR = /usr/src/linux-2.4.25/include
```

Installing the driver

To install the driver object module in the file system, become root and run:

```
# make install
# depmod -a
# insmod dpm.o [ChipSelector=0] [ChipSelector=1]
```

Note(1): For loading driver with Intel 82546 based cards, use ‘ChipSelector=1’

Note(2): For loading driver with National DP83820 based cards, use ‘ChipSelector=0’ or leave blank (default)

Note(3): ‘modprobe’ may be used in place of ‘insmod’ to resolve module dependencies.

Configure the card using your preferred configuration tool, or edit the initialization script for the interface directly. On Red Hat, the file /etc/sysconfig/network-scripts/ifcfg-eth0 might look something like this (substituting ethernet device number, example: eth0, eth1, etc):

```
DEVICE='eth1'
BOOTPROTO='none'
ONBOOT='yes'
IPADDR=192.168.0.3
GATEWAY=192.168.0.1
```

```
TYPE=Ethernet
USERCTL=no
NETMASK=255.255.255.0
NETWORK=192.168.0.0
BROADCAST=192.168.0.255
```

Note: You should also add a line to /etc/modules.conf for each interface as shown in the following example:

SAMPLE MODULES.CONF FILE

```
alias eth1 dpm
alias eth2 dpm
alias eth3 dpm
alias eth4 dpm
options dpm ChipSelector=1 IntrHoldOff=0
```

Note: In order to load the Intel driver, the "ChipSelector=1" option is required in the 'options' field of modules.conf. Please see man page for modules.conf for additional information on module configuration.

MANUAL DRIVER INSTALLATION

```
insmod ./dpm.o # to insert driver module
ifconfig eth1 192.168.0.3 # if not already configured by system
```

Once driver is inserted you may need to run "ifconfig" command or it may be run automatically by the system.

MANUAL DRIVER REMOVAL

```
ifdown eth[n] # repeat for all 'dpm' network interfaces
rmmmod dpm # to remove driver module
```

LOADING THE DRIVER FOR INTEL BASED MODELS (5261, 5262, 5164, etc.)

Note: In order to load the Intel driver, the "ChipSelector=1" option is required during the 'insmod' as follows:

```
insmod ./dpm.o ChipSelector=1
```

TUNING THE INSTALLATION

Interrupt holdoff (programmed latency)

To increase (or decrease) the value of the programmed interrupt latency, insert the module and set the "IntrHoldOff" parameter as follows:

```
insmod ./dpm.o IntrHoldOff=1 # value can be 0, 1, 2 or 3
```

Note: A value greater than 3 is not recommended.

Other driver parameters that can be set during driver load or in modules.conf:

```
MediaSpeed=10,100,1000 # Media Speed (default 1000)
SetAutoNeg=0,1 # Setting for auto negotiation (default 1=auto)
DuplexMode=0,1 # Setting for duplex (default 1=FULL)
IntrHoldOff=0,1,2 # interrupt holdoff value in 100 microsecond
units
ChipSelector=0,1 # default=0 (National), 1=Intel
NumBufDescs=<cnt> # default=256, 128, 256, 512, 1024
MaxBufferSize=<size> # default=2048, 2048, 4096, 8192
FrameGenSize=<size> # frame generator test size (64-9000 bytes)
```

See "Command Line Parameters" table below for additional information on driver module insertion parameters.

'Insmod' Command Line Parameters

The following parameters are used by entering them on the command line with the modprobe or insmod command. For example, with Intel based card model (ex. 5262, 5164, etc.) entering:

```
insmod dpm ChipSelector=1 IntrHoldOff=0
```

loads the dpm driver setting it for Intel chipset and setting the Interrupt Holdoff latency to zero (disabled).

Parameter Name	Valid Range/Settings	Default	Description
SetAutoNeg	0,1	1	This parameter specifies the enabling of auto negotiation.
DuplexMode	0,1,2 (0=half, 1=full, 2=both)	1	Defines the direction in which data is allowed to flow. Can be either one or two-directional. If both. If auto negotiation is enabled, defines modes advertised. If auto negotiation is disabled sets duplex to this value.
PauseFlowEn	0, 1	1	This parameter enables or disables pause frame flow control (typically enabled).
IntrHoldOff	0, 1 or 2 (0=off)	1	This value delays the generation of receive interrupts in units of 100 microseconds. Receive interrupt reduction can improve CPU efficiency if properly tuned for specific network traffic. Increasing this value adds extra latency to frame reception and if set too high could decrease the throughput of TCP traffic.
MediaSpeed	10, 100, 1000	1000	If auto negotiation is disabled, speed forces the line speed to the specified value in megabits per second (Mbps). If auto negotiation is enabled, sets the highest speed advertised. (adapters using copper connections only)
NumBufDescs	80 - 4096	256	This value is the number of transmit and receive descriptors allocated by the driver. Increasing this value allows the driver to queue more transmit and receive buffers. Each descriptor is 16 bytes.
MaxMtuSize	1500-9600	1500	This value is the maximum mtu size supported in the driver.
MaxBufferSize	2048, 4096, 8192, 16384	2048	Sets the maximum buffer size supported in the driver.
AccAllUni	0, 1	0	Setting this parameter to one instructs the driver

			to receive all unicast frames (sometimes useful in frame generator testing). Do not enable for normal traffic.
ChipSelector	0, 1	0	Sets the chipset selector (0=National, 1=Intel). Default is National chipset.
FrameGenSize	60-MaxMtuSize	1500	Sets the frame generator frame size for testing end-to-end or loopback.

FUNCTIONALITY TESTING

When the driver is loaded into the system via `insmod` it probes the PCI bus to locate all DP83820 devices, and creates control structures for each. The driver logs a couple of messages available in `/var/log/messages` for each device with information about its PCI geographic location, IRQ, IO address, and some basic debug information (addresses of some important structures).

All the devices on the PCI bus can be listed by,

```
# cat /proc/pci
# cat /proc/interrupts
```

IRQ and IO address information from this can be correlated with the information displayed by the driver in `/var/log/messages`

When the TCP/IP stack is initialized, it opens all configured ethernet devices, and initializes them for use. At this time, the driver will perform auto negotiation and log information about the link status. The driver can then be tested by running ping, telnet, ftp, NFS etc.

Suggested basic verifications to be run

```
lsmod           # lists loaded drivers
ifconfig        # lists configured network interfaces
dmUtil -s eth[n] # prints low-level driver statistics for device
ping <ip addr> # ping to remote station to verify
```

ADDITIONAL VERIFICATION AND PERFORMANCE TESTING

Verification and performance testing can be done using the Linux "uxBlaster" and "uxBlastee" test programs found on OEM developer kit CD. The 'dmUtil' utility program also found on the CD can be used to obtain valuable information used for debug and integration verification.

```
dmUtil -s eth0 # displays low-level driver statistics
```

```
dmUtil -m eth0 # displays DP83820 MAC controller registers
dmUtil -p eth0 # displays gigabit (phy) transceiver registers
dmUtil -e eth0 # displays eeprom
dmUtil -a eth0 # displays mac address
dmUtil -ms eth0 # displays mac stats (Intel only)
```

LOOPBACK PERFORMANCE TESTING

1. Edit 'Makefile'
2. Un-comment the following define:

```
CFLAGS += -DNS_FRAME_TEST -DIN_FRAME_TEST
```

3. Save file, unload and reload driver using procedures described above.
4. Use "ifconfig" command to bring up Ethernet interface.
5. Run the following command from the 'util' directory:

```
./dmUtil -l eth1 # loopback eth0, eth1, etc.
```

6. View statistics using dmUtil command as follows:

```
./dmUtil -s eth1 1 # second arg=1 zeroes statistics after gathering
```

Note: The driver must be compiled without "-DNS_FRAME_TEST" for normal operation.

Jumbo frames: The mtu size can be increased using the ifconfig utility, as follows:

```
# ifconfig <interface-name> mtu <mtu-size>
```

Example: ifconfig eth0 mtu 3000

Note: Use of jumbo frames is not recommended as it is not IEEE standard and many switches do not support the forwarding of jumbo frames. It also has to be enabled and negotiated on both ends of the connection and in today's faster systems, using jumbo frames offers little performance improvement and may only be useful in closed application-specific networks.

Using dmUtil dpm driver utility:

The “dpm driver utility” is used to capture detailed board levels statistics, controller registers and to set loopback mode. Example usage of the “dmUtil” are as follows:

```
dmUtil -s eth1          # get driver statistics for interface 1
dmUtil -p eth1          # get phy (transceiver) registers for interface 1
dmUtil -m eth1          # get MAC (PCI controller) registers for interface 10
dmUtil -t eth1          # get execution debug trace for interface 1
                        # (DEBUG_FLAGS trace option must be compiled in)
dmUtil -e eth1          # get eeprom configuration settings
dmUtil -ms eth1         # get mac MIB statistics
dmUtil -a eth1          # get mac address
dmUtil -w eth1 <0,1,2> # write default eeprom configuration to controller

dmUtil -l eth1 <arg>   # set controller in internal loopback mode
                        # (driver must be compiled with -DNS_FRAME_TEST
                        # in Makefile). Arg=1 for enable, 0 for disable.
```

10.2 VXWORKS DRIVER INSTALLATION AND USAGE

Driver utility shell commands for vxWorks driver operation are shown in the next section.

Please also see the vxWorks OEM Developers Guide p/n 131901 for vxWorks software installation and integration instructions.

10.3 DRIVER UTILITY COMMAND REFERENCE

10.3.1 STATISTICS SHOW FUNCTION

vxWorks usage:

inShow(unit) # unit = device index starting from 0

Linux usage:

./dmUtil -s eth1 # eth1, eth2, etc.

Command output:

```
in:   unit: 2,   ioAddr: 0xe087c000, intNum: 0x30,   checksum offload: 0, rev: 1.16m
      link up: 1, full dup: 1,   speed: 1000,   tbiMode: 0
      pci64: 1,   pci66: 1,   pci-x: 0
      max frame size: 2048,   numTxBds: 256,   intrHoldOff: 1
interrupt cnt          369621          ints/sec              28880
task cnt              369895          timer cnt             311
data timeout cnt      311          unclaimed int cnt    0
mgmt int cnt          1          RX ERR INTR CNT     0
RX OVRN INT CNT       0          RX SEQ INT CNT      0
RX SHRT INT CNT       0          RX BUF ERRS         0
RX CKSUM ERRS         0          TX ERR INTR CNT     0
TX BUF ERRS           0          PCI ERR CNT         0
rx frame cnt          1051218         last rx queued       4
max rx queued         128          tx frame cnt         1051397
tx short segs         0          tx frame segs        1051397
tx cmpl cnt           1051227         last tx queued       244
max tx queued         244          last tx cmpl         5
max tx cmpl           244          max tx segs          1
max tx frm len        1500         tx cpy cnt           0
rx byte cnt           1576827000        tx byte cnt           1577095500
tot byte cnt          3153922500        tot frame cnt         2102615
tot bytes/sec         246072000        tot frames/sec        164048
rx alloc errs         0          tx alloc errs        0
seq num errs          0          bad len cnt          0
rx flow cnt           0          bad len size         0
tx flow on cnt        8866         tx flow off cnt      8865
tx empty cnt          0          rx err bits           00000000
rx extsts bits        00000000        tx err bits           00000000
init cnt              1          link down int cnt    0
phy reset cnt         0          link up int cnt      1
rx reset cnt          0          rx start cnt         1
sig det down cnt      0          auto neg start cnt   0
sig det up cnt        0          auto neg cmpl cnt    0
phy cfg cnt           1          phy addr              1
link fail state       0          failovr chk cnt      0
recover cnt           0          test int req cnt     0
test int rsp cnt      0          ints disabled cnt    6707
re-read ints          0          num priority queues  1
tx cleanup buf cnt    0          tx loop frames dropped 0
out of buffers        0          tx flow ctrld        1
rx flow ctrld         0          tmr state             2
data timeout          0          bad len val           0
```

10.3.2 PCI REGISTERS SHOW FUNCTION

vxWorks usage:

`inShowPciRegs(unit)` # unit = device index starting from 0

Linux usage:

n/a - use Linux "lsdev" command (see man page for command 'lsdev')

Command output:

```
in: unit: 0, bus: 16, dev: 1, func: 0, pci regs:
      device&vendor: 81885120
      stsAndErrs: 02300017
      membaseCsr: 00020004
      flashbaseCsr: 00000000
      iobaseCsr: 00001001
      membaseUpper: 00000000
      irq: 00ff0106
      cfgLat/cache: 00802408
```

10.3.3 MAC REGISTERS SHOW FUNCTION

vxWorks usage:

```
inShowMacRegs(unit)    # unit = device index starting from 0
```

Linux usage:

```
./dmUtil -m eth1      # eth1, eth2, etc.
```

Command output:

```
in: unit: 0, ioAddr: 40020000, mac regs:
  CTRL: (0x0000): 0x18f01ae1 (PRST: 0, TFCE: 1, RFCE: 1, RST: 0, LRST: 0, FD: 1)
                                (FRCD: 1, FRCS: 1, SPD: 2, ILOS: 1, ASDE: 1)
  STATUS: (0x0008): 0x0000db83 (PCIX: 0, BU64: 1, PC66: 1, TBI: 0, TXOF: 0)
                                ( LU: 1,  FD: 1)

  EECD: (0x0010): 0x110
  EERD: (0x0014): 0xd8953f10
EXT-CTRL: (0x0018): 0xc00
  MDIC: (0x0020): 0x18350000
  FCAL: (0x0028): 0xc28001
  FCAH: (0x002c): 0x100
  FCT: (0x0030): 0x8808
  VET: (0x0038): 0x8100
  ICR: (0x00c0): 0x81
  ITR: (0x00c4): 0x0
  ICS: (0x00c8): 0x0
  IMS: (0x00d0): 0x0
  IMC: (0x00d8): 0x0
  RCTL: (0x0100): 0x643822a (EN: 1, BAM: 1, bsize: 3)
  FCTTV: (0x0170): 0x200
  TXCW: (0x0178): 0x1a0
  RXCW: (0x0180): 0xc000000
  TCTL: (0x0400): 0x204000a
  TIPG: (0x0410): 0x802008
  LEDCTL: (0x0e00): 0x7068e09
  PBA: (0x1000): 0x100030
  RDBAL: (0x2800): 0x81778760
  RDBAH: (0x2804): 0x0
  RDLEN: (0x2808): 0x1000
  RDH: (0x2810): 0xe0
  RDT: (0x2818): 0xd5
  RDTR: (0x2820): 0x32
  RADV: (0x282c): 0x32
  TDBAL: (0x3800): 0x81777760
  TDBAH: (0x3804): 0x0
  TDLEN: (0x3808): 0x1000
  TDH: (0x3810): 0xe2
  TDT: (0x3818): 0x88
  TIDV: (0x3820): 0x32
  TXDCTL: (0x3828): 0x1040202
  TADV: (0x382c): 0x32
  RAL[0]: (0x5400): 0xcc25000
  RAH[0]: (0x5404): 0x80001e99
  RAL[1]: (0x5408): 0xcc25000
  RAH[1]: (0x540c): 0x8000fe90
  RAL[2]: (0x5410): 0xcc25000
  RAH[2]: (0x5414): 0x8000ff90
```

10.3.4 MAC STATISTICS SHOW FUNCTION

vxWorks usage:

```
inShowMacStats(unit)    # unit = device index starting from 0
```

Linux usage:

```
./dmUtil -ms eth1      # eth1, eth2, etc.
```

Command output:

```
in: unit: 0, ioAddr: 40020000, mac regs:
  CRCERRS: (0x4000): 0
  ALGNERRC: (0x4004): 0
  SYMERRS: (0x4008): 0
  RXERRC: (0x400c): 0
  MPC: (0x4010): 0
  SCC: (0x4014): 0
  ECOL: (0x4018): 0
  MCC: (0x401c): 0
  LATCOL: (0x4020): 0
  COLC: (0x4028): 0
  DC: (0x4030): 0
  TNCRS: (0x4034): 0
  SEC: (0x4038): 0
  CEXTERR: (0x403c): 0
  RLEC: (0x4040): 0
  XON-R: (0x4048): 0
  XON-T: (0x404c): 0
  XOFF-R: (0x4050): 0
  XOFF-T: (0x4054): 0
  FC-UNSUPP: (0x4058): 0
  PKT-64: (0x405c): 0
  PKT-127: (0x4060): 0
  PKT-255: (0x4064): 0
  PKT-511: (0x4068): 0
  PKT-1023: (0x406c): 0
  PKT-MAX: (0x4070): 79354270
  GPRC: (0x4074): 79354270
  BPRC: (0x4078): 0
  MPRC: (0x407c): 0
  GPTC: (0x4080): 79354270
  OCTETS-R: (0x4088): 3384705088
  NODEF: (0x408c): 27
  OCTETS-T: (0x4090): 3384705088
  NODEF: (0x4094): 27
  NODEF: (0x40a0): 0
  NODEF: (0x40a4): 0
  NODEF: (0x40a8): 0
  NODEF: (0x40ac): 0
  NODEF: (0x40b0): 0
  NODEF: (0x40b4): 0
  NODEF: (0x40b8): 0
  NODEF: (0x40bc): 0
  NODEF: (0x40c0): 3384703584
```

NODEF: (0x40c4):	27
NODEF: (0x40c8):	3384703584
NODEF: (0x40cc):	27
NODEF: (0x40d0):	79354269
NODEF: (0x40d4):	79354270
NODEF: (0x40d8):	0
NODEF: (0x40dc):	0
NODEF: (0x40e0):	0
NODEF: (0x40e4):	0
NODEF: (0x40e8):	0
NODEF: (0x40ec):	79354269
NODEF: (0x40f0):	0
NODEF: (0x40f4):	0
NODEF: (0x40f8):	0
NODEF: (0x40fc):	0

10.3.5 PHY REGISTERS SHOW FUNCTION

vxWorks usage:

```
inShowPhyRegs(unit)      # unit = device index starting from 0
```

Linux usage:

```
./dmUtil -p eth1         # eth1, eth2, etc.
```

Command output:

```
in: unit: 0, phy regs:
    BMCR: (0x00): 0x4140
    BMSR: (0x01): 0x7949
    PHYIDR1: (0x02): 0x141
    PHYIDR2: (0x03): 0xc25
    ANAR: (0x04): 0xde1
    ANLPAR: (0x05): 0x0
    ANER: (0x06): 0x4
    ANNPTR: (0x07): 0x2001
    ANNPRR: (0x08): 0x0
    1KTCR: (0x09): 0x200
    1KSTSR: (0x0a): 0x0
    1KSCR: (0x0f): 0x3000
    STRAPREG: (0x10): 0x60
    LINK_AN: (0x11): 0x8100
    AUX_CTRL: (0x12): 0x0
    LED_CTRL: (0x13): 0x40
    INT_STATUS: (0x14): 0xd60
    INT_MASK: (0x15): 0x0
```

10.3.6 EEPROM SHOW FUNCTION

vxWorks usage:

```
inShowEeprom(unit)      # unit = device index starting from 0
```

Linux usage:

```
./dmUtil -e eth1        # eth1, eth2, etc.
```

Command output:

```
in: unit: 0, i82546 serial eeprom cfg:
  eeprom[0x00]: 5000
  eeprom[0x01]: 0cc2
  eeprom[0x02]: 1e99
  eeprom[0x03]: 0420
  eeprom[0x08]: a921
  eeprom[0x09]: 1104
  eeprom[0x0a]: 460b (INIT_CTRL_1)
  eeprom[0x0b]: 5262
  eeprom[0x0c]: dbad
  eeprom[0x0d]: 1010
  eeprom[0x0e]: 8086
  eeprom[0x0f]: 3468 (INIT_CTRL_2)
  eeprom[0x10]: 010d
  eeprom[0x11]: 1010
  eeprom[0x12]: 0000
  eeprom[0x13]: 2100
  eeprom[0x14]: 18c8 (INIT_CTRL_3-B, IntrPin: 1, TBI/Serdes: 0)
  eeprom[0x20]: c30d
  eeprom[0x21]: 7861
  eeprom[0x22]: 280c
  eeprom[0x23]: 2100
  eeprom[0x24]: 08c8 (INIT_CTRL_3-A, IntrPin: 0, TBI/Serdes: 0)
  eeprom[0x2f]: 0609 (LED2_MODE: 6, LED0_MODE: 9)
  eeprom[0x30]: 002c
  eeprom[0x31]: 4000
  eeprom[0x32]: 1107
  eeprom[0x33]: 0000
  eeprom[0x34]: 002c
  eeprom[0x35]: 4000
  eeprom[0x3f]: d895
```

10.3.7 MAC ADDRESS SHOW FUNCTION

vxWorks usage:

```
inShowMacAddr(unit)      # unit = device index starting from 0
```

Linux usage:

```
./dmUtil -a eth1        # eth1, eth2, etc.
```

Command output:

```
in: unit: 0, current (applied) mac address: 00 50 c2 0c 99 1e
```

10.3.8 BUFFER DESCRIPTOR STATUS SHOW FUNCTION

vxWorks usage:

```
inShowBdStats(unit)     # unit = device index starting from 0
```

Linux usage:

```
./dmUtil -d eth1        # eth1, eth2, etc.
```

Command output:

```
txPri: 0, bdCnt: tot: 256, actv: 176, macOwn: 169, drvOwn: 007, gaps: 0  
rxPri: 0, bdCnt: tot: 256, actv: 256, macOwn: 247, drvOwn: 009, gaps: 0
```

10.3.9 STARTING FRAME GENERATOR

vxWorks usage:

```
inStrtFrmGen(unit, frmSize)    # unit = device index starting from 0
```

Linux usage:

```
./dmUtil -l eth1 1            # eth1, eth2, etc.
```

10.3.10 STOPPING FRAME GENERATOR

vxWorks usage:

```
inStopFrmGen(unit)           # unit = device index starting from 0
```

Linux usage:

```
./dmUtil -l eth1 0           # eth1, eth2, etc.
```

NOTE: Frame generator must be compiled into Linux and vxWorks using the '-DIN_FRAME_TEST' option in the Makefile or project compiler flags.

10.3.11 SWITCH STATUS SHOW FUNCTION

Note: This function is applicable for Model 5468 GigPMC switch only.

vxWorks usage:

```
inShowSwitch(unit)      # unit = device index starting from 0
```

Linux usage:

```
./dmUtil -sw eth1      # eth1, eth2, etc.
```

Command output:

```
-> inShowSwitch

unit: 0, switch link sts: 0x39, rctl: 0x1

link: 0, upFlg: 1 (UP)
link: 1, upFlg: 0 (DOWN)
link: 2, upFlg: 0 (DOWN)
link: 3, upFlg: 1 (UP)
link: 4, upFlg: 1 (UP)
link: 5, upFlg: 1 (UP)
link: 6, upFlg: 0 (DOWN)
link: 7, upFlg: 0 (DOWN)
```

10.4 DPM DRIVER MANAGEMENT API

The dpm network driver management API is an api extension primarily used by the 'dmUtil' command line utility to enable various management functions or to acquire driver status and statistics. The typical api usage sequence is as follows:

1. User enters 'dmUtil' command and parameters as shown in the following example:

```
./dmUtil -s eth1
```
2. dmUtil parses command line and prepares an "ioctl" request block to send to driver. A pointer to a application buffer to hold the text message results (for example formatted statistics) is provided by dmUtil in the request block.
3. Driver receives ioctl request block performs action and copies result data to applications message buffer.

The following is an API description of the management ioctl commands issued to the driver from the 'dmUtil' application's viewpoint:

A. Opening a socket for management API

```
int s;  
s = socket (PF_INET, SOCK_STREAM, 0);
```

B. Create request block for driver ioctl

```
struct ifreq ifr;  
int subCmd;  
NpkUserCtl myIoc;  
char myDataBuf[MAX_LINES * MAX_LINE_LEN];  
  
/* set interface name */  
strcpy (ifr.ifr_name, "eth1");  
  
/* set ioctl sub-type */  
subCmd = DM_IOCTL_GET_STATS;  
  
/* set command argument */  
myIoc.arg1 = 0;  
  
myIoc.dataItm = (u_int) myDataBuf;  
ifr.ifr_data = (char *) &myIoc;
```

C. Issue ioctl command

```
/* issue ioctl to network driver */  
err = ioctl (s, SIOCDEVPRIVATE + subCmd, &ifr);
```

D. Check and print results

```
if (err < 0)  
{  
    /* perror (errno); */  
    usage ();  
    exit (1);  
}  
if (subCmd == DM_IOCTL_GET_TRC)  
{  
    print_trace();  
}  
else  
{  
    if ((subCmd != DM_IOCTL_SET_LOOP_MODE) &&  
        (subCmd != DM_IOCTL_PROG_EEPROM))  
        printf ("\n%s\n", myDataBuf);  
}
```

IOCTL COMMAND SUB-TYPES

```
/* get driver trace buffer */
#define DM_IOCTL_GET_TRC 1

/* get driver statistics */
#define DM_IOCTL_GET_STATS 2

/* get phy registers */
#define DM_IOCTL_GET_PHY_REGS 3

/* get mac registers */
#define DM_IOCTL_GET_MAC_REGS 4

/* set loopback mode */
#define DM_IOCTL_SET_LOOP_MODE 5
```

Additional arguments:

```
    arg1=0      Disable loopback
    arg1=1      Enable loopback

/* read eeprom */
#define DM_IOCTL_READ_EEPROM 6

/* program eeprom */
#define DM_IOCTL_PROG_EEPROM 7

/* get mac address */
#define DM_IOCTL_GET_MAC_ADDR 8

/* dump buffer descriptors */
#define DM_IOCTL_DUMP_BDS 9

/* get mac MIB statistics */
#define DM_IOCTL_GET_MAC_STATS 10
```

Please also see example code in dmUtil.c for additional information.

10.5 SWITCH CONFIGURATION / MANAGEMENT API

Note: This section is applicable for Model 5468 GigPMC switch only.

The following functions are provide a low-level access capability for reading and writing the configuration and management registers on the embedded switch:

Functions to reset and initialize switch:

```
STATUS inResetSwitchDevice(int unit, int initialReset);
```

```
LOCAL void inLiInitBcm5388(InLiDev *pLiDev);
```

Functions to read and write registers in the switch:

```
STATUS inReadSpiReg(int unit, int page, int offset, int bCnt, UINT8 *pData);
```

```
STATUS inWriteSpiReg8(int unit, int page, int offset, int val);
```

```
STATUS inWriteSpiReg16(int unit, int page, int offset, int val);
```

```
STATUS inWriteSpiReg32(int unit, int page, int offset, int val);
```

Note: The Intel device driver supporting the Model 5468 card initializes the switch by writing a default configuration during initialization. This is performed in the Intel driver initialization using the function “inLilnitBcm5388()”. This function may be modified to make changes to the default configuration. Additional switch management functions may be implemented using the low-level register access API shown. For additional information and register programming details, please contact technical support.

11. TESTING AND VERIFICATION

There are several ways to test your adapter on vxWorks and Linux platforms. This section provides *suggestions* on how you may test and verify your installation. Before you can proceed with any of the suggested tests, you must have previously configured your TCP/IP protocols and interfaces on your computer. Please refer to your online help and User Manuals for your particular system on instructions on setting up your TCP/IP environment.

- ❑ **ICMP Ping.** Ping is a simple verification method and it very useful for verification of cabling, adapter configuration and system configuration of the TCP/IP protocol software. You can also specify a “fast ping” and larger message sizes in order to more effectively test the protocols and adapter interface.
- ❑ **FTP File Transfer.** FTP is also an available method for testing your installation. FTP allows you to test by transferring files of different sizes. Using large file transfers is a good way to test medium transfer rates through the protocol stack.
- ❑ **HTTP Web Browsing.** Is a good means of testing the protocols and interface to the adapter.
- ❑ **Telnet.** Using Telnet you can log into another system and invoke commands to send a receive data. You can also test multiple connections within a single window by repetitively Telnet’ing back and forth between multiple systems.
- ❑ **Windows Explorer.** In a Windows environment, you can also use the Windows explorer to access files on other systems within your Workgroup or Domain.
- ❑ **Blaster / blastee** can be used to perform TCP throughput tests between VxWorks, Linux and Windows platforms. These test programs are included on the OEM developer CD.
- ❑ **Internal loopback:** The vxWorks and Linux drivers can be placed into internal loopback for a full-speed raw driver and controller throughput test. This is very useful for getting a baseline throughput measurement and also for verifying the robustness of the controller and driver in your system.

12. SPECIFICATIONS

Connector (Models 5261-RJ, 5262-RJ, 5164, 5468, 6267-RJ): Gigabit Ethernet (1000BaseT, 100BaseTX, 10BaseT): **RJ-45 CAT5e**

Connector (Model 5261-LC, 5262-LC, 6267-SFP fiber): SFP, SC or LC connector type for 50/125 or 62.5/125 micron multimode fiber or 8/125 singlemode fiber.

Drivers:

- Linux 2.4 (2.4.18 – 2.4.26), Linux 2.6 (all versions)
- Tornado 2.0.2, Tornado 2.2 and Tornado 2.2.1 (VxWorks 5.4/5.5/5.5.1)

Status Indicators (model 5164 PMC): Per port link/activity

Status Indicators (model 5468 PMC): Per port link/activity

Status Indicators (model 5261/5262 PMC): Per port link, duplex, transmit, receive

Status Indicators (model 6267-SFP PCI-X): Per port link, transmit, receive

Bus Interface: PCI v2.2 bus master, 32/64-bit, 33/66 MHz
PCI-X 1.0 (Model 6267-SFP)

Dimensions: (PMC Models 5164, 5261, 5262, 5468):
5.866" X 2.913

Dimensions: (PCI models 6267):
6.6" X 2.535"

PCI Power supply voltage: 5V or 3.3V (factory optioned), 5V is default

PCI signaling voltage: 5V and 3.3V

Performance Throughput (PCI 64/66, PCI-X): 245 Mbytes/sec (1.96 Gb) (full-duplex) sustained per port

Performance Throughput (PCI 32/33): 118 Mbytes/sec (944 Mb) full or half duplex sustained per port

Maximum frame rate: Over 850,000 frames per second sustained per port

Burst Rate: Up to 256 dwords (1024 bytes) over PCI bus

Host Offloading: IPv.4 checksum (UDP, TCP and IP) and optional statistics gathering for RFC 1213 (MIB II), RFC 1398 (Ether-like MIB), IEEE 802.3 LME

Optional: Jumbo packets, 802.3x full duplex flow control with automatic pause and priority with multiple priority queues

Link Quality Monitor: Continuously adapts to actual line conditions by managing echo and crosstalk cancellation, equalization, timing and skew compensation. Automatic Gain Control maximizes signal strength

Environmental Range:

Operating Temperature: **0° to 60°C**

Relative Humidity: 10% to 90%, non-condensing

Voltage: 5 or 3.3 volts

MTBF:

Model 5261 – 350,000 hours

Model 5262 – 300,000 hours

Model 5262 – 300,000 hours

Model 5164 – 275,000 hours

Model 5468 – 250,000 hours

Model 6267 – 350,000 hours

Environmental Standards Compliance (pending):

- FCC Part 15, Class B
- EN 55022; 1998 Class B
- EN 50082-1
- CE Mark

Standards Compliance:

- IEEE 802.3-2002 (all applicable sections for 1000 base-T, 1000 base-SX, 1000 Base-LX)
- Network: IEEE 802.3u Auto Negotiation and parallel detection
- IEEE 802.3ab Gigabit Ethernet over 4 pairs of UTP Category 5e (1000BaseT) Gigabit
- IEEE 802.3z Gigabit Ethernet over 1000 Base SX multimode fiber
- IEEE 802.1D and 802.1Q as applicable for VLAN priority queuing
- IEEE 802.3 Gigabit Ethernet over 4 pairs of UTP Category 5e 1000BaseTX).
- IEEE 802.3u Fast Ethernet over 2 pairs of UTP Category 5 (100BaseTX).
- IEEE 802.3 Ethernet over 2 pairs of UTP Category 3 (10BaseT).

- IEEE 1386 Draft 2.2 PMC Mezzanine Standard

Full Duplex: Support for 10/100/1000 Mbps data rates on all copper models (fiber models Gigabit mode only)

Virtual Network: Virtual LAN (VLAN) tag support

Distance (Models 5261, 5262, 5164, 5468 and 6267-RJ): Recommended maximum distance is 328 feet (100 meters).

12.1 ENVIRONMENTAL SPECIFICATIONS

Rugged Class	Grade	Operating Temp.	Storage Temp	Vibration	Shock	Humidity	Other Specs
C1	Commercial	0°C to +65°C 200 linear ft/minute air flow Storage temp: -50C to +100C	-40C to +70C	N/A	N/A	Operating: Up to 90% Non- Condensing	Conformal coated: No Altitude: 33,000 ft.
R1	Rugged, Forced Air	-20°C to +75°C 350 linear ft/minute air flow Storage temp: -50C to +100C	-50C to +85C	5Hz-2000Hz at 2g, 0.38mm peak displacement (operating) 5Hz-2000Hz at 5g, 0.76mm peak displacement (non- operating) Per MIL- STD-810E	20g, 11ms, ½ sine (operating); 30g, 11ms, ½ sine (non- operating) Per MIL-STD- 810E	Operating: Up to 95% Non- Condensing	Conformal coated: Yes Altitude: 50,000 ft.
R2	Rugged, Forced Air	-40°C to +85°C 450 linear ft/minute air flow Storage temp: -50C to +100C	-50C to +85C	5Hz-2000Hz at 2g, 0.38mm peak displacement (operating) 5Hz-2000Hz at 5g, 0.76mm peak displacement (non- operating) Per MIL- STD-810E	30g, 11ms, ½ sine (operating); 40g, 11ms, ½ sine (non- operating) Per MIL-STD- 810E	Operating: Up to 95% Non- Condensing	Conformal coated: Yes Altitude: 50,000 ft.

Note: Conformal coating is MIL-I-46058 compliant (typically type UR or AR) or other as applicable and based on customer requirement.

13. WARRANTY AND SUPPORT INFO

Technical Support and Warranty:

Telephone technical support (Mon-Fri 8AM to 6PM, MST), 24-hour support via web email

1 year limited product warranty on controller hardware (see warranty info)

Contacting Us

You may contact DSS Networks in one of several ways: via the Web, e-mail, fax or telephone.

Technical Support

Send all technical support queries to support@dssnetworks.com or visit the DSS Networks website at www.dssnetworks.com. The DSS Networks website contains technical as well as sales literature for all of our products.

Technical Support-Worldwide

+1.949.716.9051

Technical Support-Fax

+1.949.716.9052

Main Corporate Telephone Numbers

949-716-9052