PCI FOI

1.25 GBaud Fiber Optic Interface for PCI Local Bus

USER'S GUIDE

008-01089-01

The information in this document is subject to change without notice and does not represent a commitment on the part of Engineering Design Team, Inc. The software described in this document is furnished under a license agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of the agreement.

Engineering Design Team, Inc. ("EDT"), makes no warranties, express or implied, including without limitation the implied warranties of merchantibility and fitness for a particular purpose, regarding the software described in this document ("the software"). EDT does not warrant, guarantee, or make any representations regarding the use or the results of the use of the software in terms of its correctness, accuracy, reliability, currentness, or otherwise. The entire risk as to the results and performance of the software is assumed by you. The exclusion of implied warranties is not permitted by some jurisdictions. The above exclusion may not apply to you.

In no event will EDT, its directors, officers, employees, or agents be liable to you for any consequential, incidental, or indirect damages (including damages for loss of business profits, business interruption, loss of business information, and the like) arising out of the use or inability to use the software even if EDT has been advised of the possibility of such damages. Because some jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitations may not apply to you. EDT's liability to you for actual damages for any cause whatsoever, and regardless of the form of the action (whether in contract, tort [including negligence], product liability or otherwise), will be limited to $50.

No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, without the express written agreement of Engineering Design Team, Inc.

© Copyright Engineering Design Team, Inc. 1997–99. All rights reserved.

Sun, SunOS, SBus, SPARC, and SPARCstation are trademarks of Sun Microsystems, Incorporated.

Windows NT is a registered trademark of Microsoft Corporation.

Intel and Pentium are registered trademarks of Intel Corporation.

UNIX is a registered trademark of X/Open Company, Ltd.

OPEN LOOK is a registered trademark of UNIX System Laboratories, Inc.

Red Hat is a trademark of Red Hat Software, Inc.

IRex is a trademark of Silicon Graphics, Inc.

AIX is a registered trademark of International Business Machines Corporation.

Xilinx is a registered trademark of Xilinx, Inc.

Kodak is a trademark of Eastman Kodak Company.

The software described in this manual is based in part on the work of the independent JPEG Group.

EDT and Engineering Design Team are trademarks of Engineering Design Team, Inc.

Contents

Tables

Overview

The PCI Bus Remote Camera Fiber Optic Interface (PCI FOI) is a PCI local bus direct memory access (DMA) board, resident in a host computer running either UNIX-based or Windows NT operating systems. The PCI FOI implements a high-speed fiber optic interface between the host system and a remote camera interface module (PCI RCI). The remote camera interface module connects the fiber optic cable to the parallel interface of the selected digital camera. A complete PCI FOI Remote Camera Interface System consists of this PCI DMA fiber optic interface board and one to four remote camera interface modules.

The complete PCI RCI system, including the remote unit, the fiber optic interface, and the digital video camera software, is described in three manuals:

- The *PCI RCI Remote Camera Interface User's Guide*, EDT part number 008-01091, describes the remote camera interface module.
- This manual, the *PCI FOI 1.25 GBaud Fiber Optic Interface for PCI Local Bus User's Guide*, EDT part number 008-01089, describes the PCI FOI board in the host computer and the generic software that comes with it.
- The *PCI DV Digital Video Camera Interface User's Guide*, EDT part number 008-00966, describes the software that operates digital video cameras connected to the remote unit. The manual also describes the PCI DV hardware; this portion is irrelevant to the PCI RCI system.

This manual describes the operation of the PCI FOI with UNIX-based and Windows NT operating systems. It is divided into the following sections:

After Installing	gives instructions for verifying the installation, building the sample programs, uninstalling the software, and upgrading the firmware when necessary.
Input and Output	describes the programming library calls.
Gigabit Fiber Optic Interface Protocol	describes the PCI FOI signals.
Registers	describes the hardware registers.
Specifications	lists the product specifications.

After Installing

After installing the PCI FOI, verify the installation and build the sample programs, if you wish. Instructions for uninstalling the software and upgrading the firmware are also provided if necessary.

Verifying the Installation

To verify that installation was successful and that the PCI FOI is operating correctly, run the following test with self-generating data on the interface Xilinx:

1. If you're using Windows NT, double-click on the Pcd Utilities shortcut.

   If you're using a UNIX-based system, navigate to the directory in which you installed the driver.

2. At the prompt, enter:

   ```
   xtest 4096
   ```

   The PCI FOI returns test status information. The following is an example of proper behavior, although details will vary:

   ```
   reading 4096 words
   buf at 820000
   testing dirreg at 4 4
   testing dirout at 8 8
   testing dirin at 8 c
   testing ctlout at a a
   testing ctlin at a e
   Calling DMA read 8192 at 820000
   return to do read:
   read returned length 8192
   Done.
   checking data
   4096 words 0 errors
   buf 0 820000
   buf 1 920000
   reading 100 buffers of 1048576 bytes from unit 0 with 2 bufs
   return to start: starting read at 820000
   starting read at 920000
   hit return to continue:
   counter0 4628 2362958141 counter1 4628 3122437420 freq 0 266230000
   dtime 759479279.000000 ticktime 266230000.000000
   time is 2.852719 sec
   36757077.671371 bytes/sec
   ```

Building the Sample Programs

UNIX-based Systems

To build any of the example programs on UNIX-based systems, enter the command:

```
make file
```

where *file* is the name of the example program you wish to install.

To build and install all the example programs, enter the command:

```
make
```

All example programs display a message that explains their usage when you enter their names without parameters.

Windows NT Systems

To build any of the example programs on Windows NT systems:

1. Run Pcifoi Utilities.
2. Enter the command:

```
nmake file
```

where *file* is the name of the example program you wish to build.

To build and install all the example programs, simply enter the command:

```
nmake
```

All example programs display a message that explains their usage when you enter their names without parameters.

You can also build the sample programs by setting up a project in Windows Visual C++.

Uninstalling

UNIX-based Systems

To remove the PCI FOI driver on UNIX-based systems:

1. Become root or superuser.
2. Enter:

```
pkgrm EDTpcifoi
```

For further details, consult your operating system documentation, or call Engineering Design Team.

Windows NT Systems

To remove the PCI FOI toolkit on Windows NT systems, use the Windows NT Add/Remove utility. For further details, consult your Windows NT documentation.

You can always get the most recent update of the software from our web site, **www.edt.com**. See the document titled *Contact Us*.

Upgrading the Firmware

After upgrading to a new device driver, it may sometimes also be necessary to upgrade the PCI interface PROM. If so, the *readme* file will say so.

To do so, download firmware to the board's PCI interface Xilinx using the *pciload* program:

1. If you're using Windows NT, double-click on the Pcd Utilities shortcut.

   If you're using a UNIX-based system, navigate to the directory in which you installed the driver.

2. At the prompt, enter:

   `pciload -u unit_no` *`file.bit`*

   where *`file.bit`* is the appropriate file for the device. For example, if you have a PCI CD-60, enter:

   `pciload -u 0 pcd60.bit`

3. Shut down the operating system and turn the host computer off and then back on again. The board reloads this from flash only during power-up. Therefore, after running *pciload*, the new bit file is not in the Xilinx until the system has been power cycled. Simply rebooting is not adequate.

NOTE If the firmware becomes corrupted for some reason (for example, an incorrect file being used as input), the board could cause the system to hang when booting. If this happens, use the following procedure to recover:

---

1. Halt the system and power it off.
2. Install a jumper on the board's only two-pin jumper.
3. Power the system back on.
4. With the power still on, remove the jumper, taking care not to short out anything on the board or adjacent boards.
5. When the system reboots, perform the firmware upgrade as described above, leaving the jumper off.

Programming Interface

Tp program an application for the PCI FOI, use either the library of software routines described in the *PCI DV Digital Video Camera Interface User's Guide*, or those described in the *PCI CD Configurable DMA Interface User's Guide.*In addition, the following routines are available:

Routine	Description
edt_set_foiunit	Specifies which remote acmera interface unit on the fiber optic loop to address and acquire from.
edt_get_foiunit	Returns the currently selected remote camera interface unit.
edt_get_foicount	Returns the number of remote camera interface units on the fiber optic loop.

Table 1. PCI FOI-specific Library Routine

edt_set_foiunit

Description

Specifies which remote camera interface unit on the fiber optic loop to address and acquire data from. Units are addressed starting at 0, which is the default.

Syntax

```
#include "edtinc.h"
int edt_set_foiunit(EdtDev *edt_p, int val);
```

Arguments

edt_p	device handle returned from *edt_open*
val	the number of the unit to address and acquire from

Return

0 on success; –1 on error.

edt_get_foiunit

Description

Returns which remote camera interface unit on the fiber optic loop is currently being addressed, and from which data is currently being acquired, if acquisition is occurring. Units are addressed starting at 0.

Syntax

```
#include "edtinc.h"
int edt_get_foiunit(EdtDev *edt_p);
```

Arguments

edt_p	device handle returned from *edt_open*

Return

The currently selected remote camera interface unit.

edt_get_foicount

Description

Returns the number of remote camera interface units currently on the fiber optic loop.

Syntax

```
#include "edtinc.h"
int edt_get_foicount(EdtDev *edt_p);
```

Arguments

edt_p	device handle returned from *edt_open*

Return

The number of remote camera interface units currently on the fiber optic loop.

Gigabit Fiber Optic Interface Protocol

The EDT gigabit fiber optic interface protocol uses features provided by the Gigabit Ethernet MAC sublayer as implemented in the SEEQ 8100 Gigabit Ethernet Media Access Controller (MAC 8100) chip.

The EDT gigabit fiber optic interface protocol uses a ring topology. The ring has one PCI FOI board in the host computer serving as master, and from one to fifteen remote nodes. These nodes could be EDT PCI remote camera interface boards; however, for most cameras and applications, the aggregate bandwidth is adequate for at most four remote nodes.

All remote nodes pass all packets received on to the next node unchanged, even the remote node that is the targeted recipient of the packet. This is called *automatic pass-through*. Remote nodes can also insert new packets into the stream destined for the master. The master can address packets to a specific remote node—numbered 0 through 14—or broadcast to all remote nodes at once. The broadcast unit number is 15.

The standard Ethernet packet fields of Destination Address, Source Address, and Length/Type are not implemented. The protocol described in this section starts with the first 32-bit word of the packet. All packets have an integral number of 32-bit data words.

Each packet starts with a routing byte that specifies the kind of packet it is, and the remote node unit number for which it is targeted. You can send three kinds of packets:

- *DMA packets* send data between the host computer's memory and the remote nodes.
- *Command packets* write to the command stream in order to implement high-level control functions.
- *Register write packets* write data to hardware registers in either the remote node or the PCI FOI.

The MAC 8100 is always configured for little-endian byte ordering: the least significant bits reside in the lowest address. However, you can configure nodes to pack the 32-bit data words in DMA packets in any manner required. Accesses of PCI FOI MAC 8100 registers on big-endian machines are byte-swapped.

Routing Byte

The first byte of each packet is the routing byte. It is defined as shown in Table 2:

Bits	Name	Description
D7		reserved
D6–5	FCTN	These two bits specify the type of the packet: 00 DMA transfer 01 Command 10 Register 11 reserved
D4	DIRIN	A value of 0 indicates that the PCI FOI is the source of the packet. A value of1 indicates that the remote node is the source.
D3–0	UNIT	A value between 0 and 14 determines which of up to 15 remote nodes is the source or destination of the packet. A value of 15 is the broadcast address, indicating that all remote nodes are targeted.

Table 2. Routing Byte Format

DMA Packets

Use DMA packets to send data to or from the remote nodes.

Bits 7–0 of the first 32-bit word constitute the routing byte. The other three bytes are reserved for future uses. All subsequent 32-bit words until the end of the packet have DMA data. DMA packets typically consist of 256 bytes (64 words) of DMA data.

Register Write Packets

Register write packets write data into hardware registers of the specified node. You can use them, for example, to implement interrupts and flow control.

Only the first byte of each 32-bit word is used. Data in the other three bytes is reserved for future use. The first word contains the routing byte, the second word contains an 8-bit register address, and the third word contains a data byte.

Command Packets

You can use command packets to implement higher-level control functions. The data in command packets is typically interpreted by processors on each end.

Only the first byte of each 32-bit word is used. Data in the other three bytes is reserved for future use. All command packets are 16 words long, including the routing byte. Not all of these words are used; the first unused word is marked by a one in bit 8. The processor receives a routing byte followed by from 0–14 data bytes.

Remote nodes send a command packet only in response to a command packet from the PCI FOI on the host. This ensures that the command FIFO on the PCI FOI does not overflow, even when multiple remote nodes are installed. The PCI FOI must not send a second command packet to a remote node until it has received a response from the previous command packet.

Command packets sent from the PCI FOI to a remote node consist of a routing byte, an opcode, and from 0–13 argument bytes. The opcode is an ASCII character. The arguments are also typically composed of ASCII characters, using hexadecimal notation for numerical values and spaces for delimiters.

Command packets sent in response by the remote node consist of a routing byte and 0–14 argument bytes. Again, the data field is typically composed of ASCII characters. If additional response data is expected, the host can send the **m** opcode repeatedly to request the next 14 bytes of response data, until fewer than 14 response bytes are returned.

The PCI FOI can send a variety of command opcodes. Table 3 shows some examples:

Op-code	Argu-ments	Description
w	addr data	Write the data to a processor address on the remote module.
r	addr	Read from a processor address on the remote module.
i		Initialize all registers on the remote module.
v		Request remote hardware and software version.
m		Request more response data from the remote module.
A		Automatic configuration: turns off pass-through mode in remote module.
U	unit	Unit assign: set the unit number and turn on pass-through in remote module.

Table 3. Command Opcode Examples

Interrupts

The PCI FOI implements hardware registers to support interrupts from the remote nodes.

The PCI FOI must implement a hardware register at address 0x0F. Any remote node can write a data value of 0x01 to this register address at any time to ask the PCI FOI within the host to reconfigure the ring of remote devices. This is called an AUTOCFGREQ interrupt, and is used to support automatic configuration.

The PCI FOI can support general purpose interrupts from each of the remote nodes. The PCI FOI can treat a value of 0x02 written to a PCI FOI register address as an interrupt request (INTERRUPTREQ). The PCI FOI determines which remote node is requesting the interrupt by checking the register address: the address must be between 0 and 14, corresponding to the remote node's unit number. The PCI FOI can service the interrupt through command packet writes to the remote node, in a manner specified by the remote node documentation. Only after the interrupt has been serviced can the remote node initiate another interrupt.

Automatic Configuration

At reset, all remote nodes are configured to pass any packets received on to the next node in the ring. Also at reset, all remote nodes will send an AUTOCFGREQ interrupt to the PCI FOI host approximately once per second.

Whenever the PCI FOI host receives an AUTOCFGREQ register write, it responds by sending an AUTOCONFIG command packet with an opcode of **A** to the broadcast unit number 15. This tells all remote nodes to disable the automatic pass-through of all received packets.

When this AUTOCONFIG command packet is received back at the host after its trip around the ring, the host sends a UNITASSIGN command packet with an opcode of **U** and a numeric argument of 0 (in ASCII) to the broadcast unit number 15.

The first remote node takes the argument of zero for its unit number, increments the argument by one, passes the UNITASSIGN command packet on to the next node, and turns automatic pass-through back on. Each subsequent node does the same. After receiving the AUTOCONFIG command packet, therefore, the PCI FOI must wait 10 ms before responding with UNITASSIGN, in order to give each remote node enough time to turn off automatic pass-through.

When the UNITASSIGN command packet completes its circle around the ring and returns to the PCI FOI, the value of the argument equals the number of remote nodes in the ring. The maximum legal value for the UNITASSIGN argument received by the PCI FOI is 15, indicating 15 remote nodes in the loop numbered 0 through 14. The autoconfiguration sequence is then complete.

Because the ultimate target of both the AUTOCONFIG and the UNITASSIGN command packets is the PCI FOI itself, these packets are sent with the DIRIN bit of the routing byte set. Remote nodes accept any command packet with the broadcase unit number of 0x0F regardless of the state of the DIRIN bit.

If a break is found in the ring, the AUTOCONFIG opcode does not reach some o f the remote nodes, and they continue emitting register writes to the PCI FOI's INTERRUPTREQ register. The PCI FOI must then report how many such writes occur each second, so you can discover where the break is.

Error Recovery

On all nodes, any received packet that shows an error through the MAC 8100 Receive_Discard pin is discarded. The PCI FOI recovers from errors in command packets by having the host timeout if no response is received from the remote node. Errors in DMA packets result in missing blocks of data. Errors in register write packets result in no action taken.

Registers

The PCI FOI has two memory spaces: the memory-mapped registers and the configuration space. Expansion ROM and I/O space are not implemented. Applications can access the PCI FOI registers through the DMA library routines, or if necessary by means of *ioctl( )* calls with PCI FOI-specific parameters, as defined in the file *pcifoi.h*.

Configuration Space

The configuration space is a 64-byte portion of memory required to configure the PCI Local Bus and to handle errors. Its structure is specified by the PCI Local Bus specification. The structure as implemented for the PCI FOI is as shown in Figure 1 and described below.

Address Bits	31 16	15 0
0x00	Device ID = 0x30	Vendor ID = 0x123D
0x04	Status *(see below)*	Command *(see below)*
0x08	Class Code = 0x088000	Revision ID = 0 *(will be updated)*
0x0C	BIST = 0x00 / Header Type= 0x00	Latency Timer *(set by OS)* / Cache Line Size *(set by OS)*
0x10	Base Address Register *(set by OS)*	
	not implemented	
0x3C	Max_Lat = 0x04 / Min_Gnt = 0x04	Interrupt Pin = 0x01 / Interrupt Line *(set by OS)*

Figure 1. Configuration Space Addresses

Values for the status and command fields are shown in Tables 4 and 5. For complete descriptions of the bits in the status and command fields, see the *PCI Local Bus Specification*, Revision 2.1, 1995, available from:

PCI Special Interest Group
P.O. Box 14070
Portland, OR 97214
Phone: 800/433-5177 (United States) or 503/797-4207 (international)
Fax: 503/234-6762

Bit	Name	Value	Bit	Name	Value
0–4	reserved	0	10	DEVSEL Timing	0
5	66 MHz Capable	0	11	Signaled Target Abort	implemented
6	UDF Supported	0	12	Received Target Abort	implemented
7	Fast Back-to-back Capable	0	13	Received Master Abort	implemented
8	Data Parity Error Detected	implemented	14	Signaled System Error	implemented
9	DEVSEL Timing	1	15	Detected Parity Error	implemented

Table 4. Configuration Space Status Field Values

Bit	Name	Value	Bit	Name	Value
0	IO Space	0	6	Parity Error Response	implemented
1	Memory Space	implemented	7	Wait Cycle Control	0
2	Bus Master	implemented	8	SERR# Enable	implemented
3	Special Cycles	0	9	Fast Back-to-back Enable	implemented
4	Memory Write and Invalidate Enable	0	10–15	reserved	0
5	VGA Palette Snoop	0			

Table 5. Configuration Space Command Field Values

PCI Local Bus Addresses

Figure 2 describes the PCI FOI interface registers in detail. The addresses listed in Figure 2 are offsets from the gate array boot ROM base addresses. This base address is initialized by the PCI Local Bus host operating system at boot time.

NOTE The addresses 0x80 and 0x84 are used by the pciload utility to update the gate array. User applications must not modify use these registers. Results of running pciload do not take effect until after the board has been turned off and then on again.

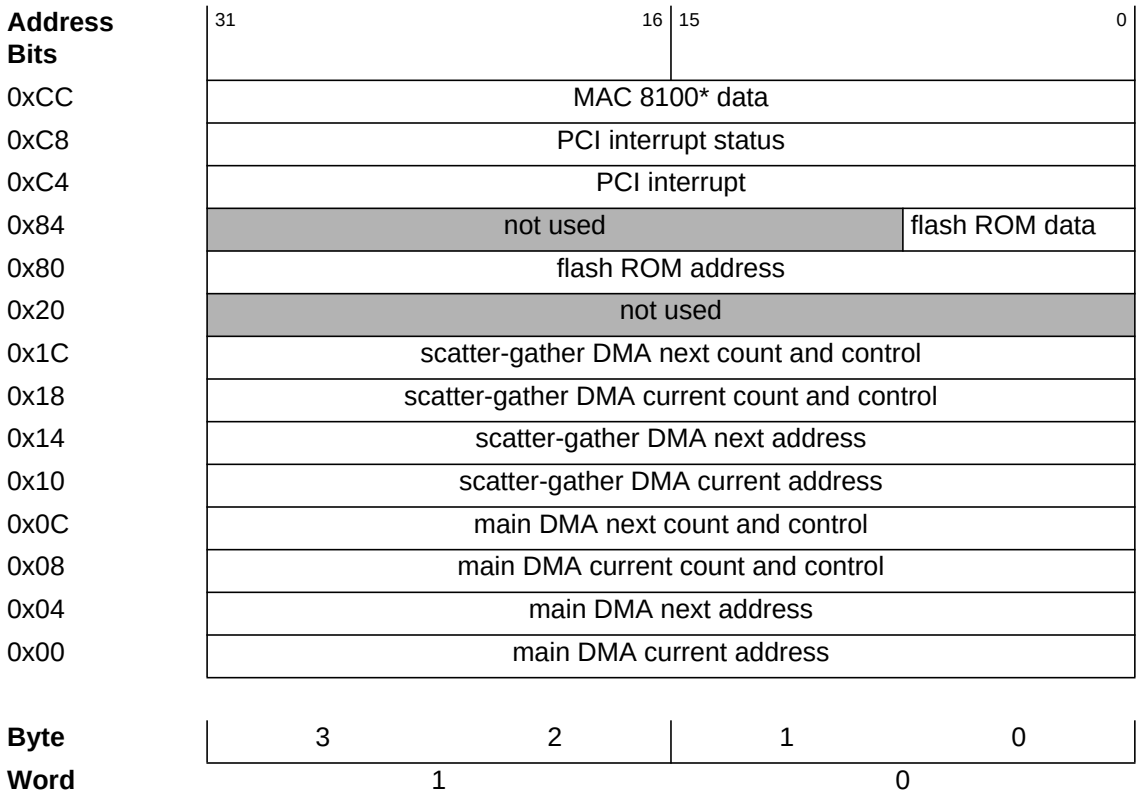

Figure 2. PCI Local Bus Addresses

*The Mac 8100 is the SEEQ 8100 Gigabit Ethernet Media Access Controller. For details of its design and operation, see www.seeq.com.

Scatter-gather DMA

PCI Direct Memory Access (DMA) devices in Intel-based computers access memory using physical addresses. Because the operating system uses a memory manager to connect the user program to memory, memory pages that appear contiguous to the user program are actually scattered throughout physical memory. Because DMA accesses physical addresses, a DMA read operation must *gather* data from noncontiguous pages, and a write must *scatter* the data back to the appropriate pages. The PCI FOI driver uses information from the operating system to accomplish this. The operating system passes the driver a list of the physical addresses for the user program memory pages. With this information, the driver builds a scatter-gather (SG) table, which the DMA device uses sequentially.

Most other PCI computers offer memory management for the PCI bus as well, so the operating system needs to pass only the address and count for DMA. The addresses appear contiguous to the PCI bus.

The scatter-gather DMA list is stored in memory. The scatter-gather DMA channel copies it as required into the main DMA registers. The format of the DMA list in memory is as follows (illustrated in Figure 3):

- Each page entry takes eight bytes. Therefore, the scatter-gather DMA count is always evenly divisible by eight.
- The first word consists of the 32-bit start address of a memory page.
- The most significant 16 bits of the second word contain control data.

- The least significant 16 bits of the second word contain the count.

As of the current release, only bit 16 contains control information. When set to one, and when enabled by setting bit 28 of the Scatter-gather DMA Next Count and Control register, this bit causes the main DMA interrupt to be set when the marked page is complete.

Bits	63 32	31	16	0
Each entry	address	control (unused)	DMA int	count

Figure 3. Scatter-gather DMA List Format

Performing DMA

All main DMA registers are read-only. Only the corresponding scatter-gather DMA registers must write to them. To initiate a DMA transfer:

1. Set up one or more scatter-gather DMA lists in host memory, using the format described above and illustrated in Figure 3.
2. Write the address of the first entry in the list to the Scatter-gather Next DMA Address register.
3. Write the length of the scatter-gather DMA list to the Scatter-gather Next DMA Count and Control register, setting the interrupts as you require. Ensure that bit 29 of this register is set to 1: this starts the DMA.
4. If the DMA list is greater than one page, load the address of the first entry of the next page and its length, as described in steps 2 and 3, when bit 29 of the Scatter-gather Next DMA Count and Control register is asserted.

Main DMA Current Address Register

Size	32-bit
I/O	read-only
Address	0x00
Access	`EDT_DMA_CUR_ADDR`
Comments	Automatically copied from the main DMA next address register after main DMA completes.

Bit	Description
A31–0	The address of the current DMA, or the last used address if no DMA is currently active.

Main DMA Next Address Register

Size	32-bit
I/O	read-only
Address	0x04
Access	EDT_DMA_NXT_ADDR
Comments	The scatter-gather DMA fills this register when required from the scatter-gather DMA list..

Bit	Description
A31–0	Read the starting address of the next DMA.

Main DMA Current Count and Control Register

Size	32-bit
I/O	read-only
Address	0x08
Access	EDT_DMA_CUR_CNT
Comments	This register is automatically copied from the main DMA next count and control register after main DMA completes.

Bit	Description
D31-16	Read-only versions of bits 31–16 of the Scatter-gather DMA current count and control register.
D15–0	The number of words still to be transferred in the current DMA.

Main DMA Next Count and Control Register

Size	32-bit
I/O	read-only
Address	0x0C
Access	EDT_DMA_NXT_CNT
Comments	The scatter-gather DMA fills this register when required from the scatter-gather DMA list.

Bit	Description
D31-16	Read-only versions of bits 31–16 of the Scatter-gather DMA next count and control register.
D15–0	The number of words still to be transferred in the current DMA.

Scatter-gather DMA Current Address Register

Size 32-bit

I/O read-only

Address 0x10

Access EDT_SG_CUR_ADDR

Comments Automatically copied from the scatter-gather DMA next address register when that register is valid and the current scatter-gather DMA completes.

Bit	Description
A31–0	The address of the current DMA, or the last used address if no DMA is currently active.

Scatter-gather DMA Next Address Register

Size 32-bit

I/O read-write

Address 0x14

Access EDT_SG_NXT_ADDR

Comments The driver software writes this register as described in step 2 of the list in the section entitled "Performing DMA" on page 13.

Bit	Description
A31–0	The starting address of the next DMA.

Scatter-gather DMA Current Count and Control Register

Size 32-bit

I/O read-only

Address 0x18

Access EDT_SG_CUR_CNT

Comments The driver software can read this register for debugging or to monitor DMA progress.

Bit	Description
D31-16	Read-only versions of bits 31–16 of the Scatter-gather DMA next count and control register.
D15–0	The number of words still to be transferred in the current DMA.

Scatter-gather DMA Next Count and Control Register

Size 32-bit

I/O read-write

Address 0x1C

Access EDT_SG_NXT_CNT

Comments The driver software writes this register as described in step 3 of the list in the section entitled "Performing DMA" on page 13.

Bit	EDT_	Description
D31	EN_RDY	Enable scatter-gather next empty interrupt. A value of 1 enables DMA_START (bit 29 of this register) to set DMA_INT (bit 12 of the Status register), thus causing an interrupt if the PCI_EN_INTR bit is set (bit 15 of the Main DMA Command and Configuration register). A value of 0 disables the DMA_START from causing an interrupt.
D30	DMA_DONE	Read-only: a value of 0 indicates that a scatter-gather DMA transfer is currently in progress. A value of 1 indicates that the current scatter-gather DMA is complete.
D29	DMA_START	Write a 1 to this bit to indicate that the values of this register and the SG DMA Next Address register are valid; this sets this bit to 0, indicating either that the copy is in progress, or that the device is waiting for the current DMA to complete. In either case, this register and the SG DMA Next Address register are not available for writing. Reading a value of 1 indicates that the SG DMA Next Count and SG DMA Next Address registers have been copied into the SG DMA Current Count and SG DMA Current Address registers and that the Next Count and Next Address registers are once more available for writing.
D28	EN_MN_DONE	A value of 1 enables the main DMA page done interrupt (bit 18).
D27	EN_SG_DONE	Enable scatter-gather DMA done interrupt. A value of 1 enables DMA_DONE (bit 30 of this register) to set DMA_INT (bit 12 of the Status register), thus causing an interrupt if the PCI_EN_INTR bit is set (bit 15 of the Main DMA Command and Configuration register). A value of 0 disables the DMA_DONE from causing an interrupt.
D26	DMA_ABORT	A value of 1 stops the DMA transfer in progress and cancels the next one, clearing bits 29 and 30. Always 0 when read.
D25	DMA_MEM_RD	A value of 1 specifies a read operation; 0 specifies write.
D24	BURST_EN	A value of 0 means bytes are written to memory as soon as they are received. A value of 1 means bytes are saved to write the most efficient number at once.
D23	MN_DMA_DONE	Read only: a value of 1 indicates that the main DMA is not active.
D22	MN_NXT_EMP	Read only: a value of 1 indicates that the main DMA next address and next count registers are empty.
D21–19		Reserved for EDT internal use.

Bit	EDT_	Description
D18	PG_INT	Read-only: a value of 1 indicates that the page interrupt is set (enabled by bit 28 of this register), and that the main DMA has completed transferring a page for which bit 16 (the page interrupt bit) was set in the scatter-gather DMA list (see Figure 3). If the PCI interrupt is enabled (bit 15 of the PCI interrupt and remote Xilinx configuration register), this bit causes a PCI interrupt. Clear this bit by disabling the page done interrupt (bit 28 of this register).
D17	CURPG_INT	Read-only: a value of 1 indicates that bit 16, the page interrupt bit, was set in the scatter-gather DMA list entry for the current main DMA page.
D16	NXTPG_INT	Read-only: a value of 1 indicates that bit 16, the page interrupt bit, was set in the scatter-gather DMA list entry for the next main DMA page.
D15–0		The number of bytes in the next scatter-gather DMA list.

Flash ROM Access Registers

Flash ROM Address Register

Size 32-bit

I/O read-write

Address 0x80

Comments Use this register and the flash ROM data register (below) to update the program in the field-programmable gate array that implements the PCI interface.

Bit	Description
D31–25	Reserved for EDT internal use.
D24	A value of 1 causes the data in the flash ROM data register to be written to the address specified by bits 0 through 23. A value of 0 reads the data.
D23–0	Address of location in flash ROM that the next read or write will access.

Flash ROM Data Register

Size 32-bit

I/O read-write

Address 0x84

Comments Use this register and the flash ROM address register (above) to update the program in the field-programmable gate array that implements the PCI interface.

Bit	Description
D31–9	not used
D8	A read-only bit indicating the position of the jumper that enables access to the protected area of the ROM that contains the executable program. A value of 1 indicates that the board can load a new program.
D7–0	The new data to load into flash ROM with a write operation (specified by setting bit A24 in the flash ROM address register), or the data that was read (specified by clearing bit A24 in the flash ROM address register).

Other Registers

Fiber Message Interface Register

Size 32-bit

I/O read-write

Address 0xC0

Comments Intended to end short messages to and from the remote module. Under most circumstances, user applications can rely on the driver to handle this.

You can write all bytes independently. Read bits 16–23 together as a byte, and with care, because reading them advances the read FIFO.

To send a message, write bytes into the transmit FIFO while monitoring its state. The entire message must be less than 16 bytes. When the entire message is in the FIFO, write a one to bit 8 of this register to send the message. Then monitor bit 8 to determine when the message has completed. To flush both the transmit and receive FIFOs, write a one to bit 9.

Bit	EDT_	Description
D31	FOI_DATA_AVAIL	A value of 1 indicates that more data is available from the receive message FIFO.
D30–24		not used
D23–16	FOI_RD_DATA	Read the current byte of the receive message FIFO. Then read bit 31 of this register; if its value is 1, more data is available. Read these eight bits again for the next byte of data.
D15–14		Reserved for EDT internal use.
D13	FOI_TX_FIFO_EMP	A value of 1 indicates that the transmit register file is empty.
D12	FOI_TX_FIFO_FULL	A value of 1 indicates that the transmit register file is full.
D10–11		not used

Bit	EDT_	Description
D9	FOI_FIFO_FLUSH	Flush transmit and receive FIFOs. (Always 0 when read.)
D8	FOI_MSG_SEND (write) FOI_MSG_BUSY (read)	Write a 1 to transmit the data in the transmit FIFO. Reading a value of 1 indicates that the transmit is ongoing (busy).
D 7–0	FOI_WR_DATA	Write transmit message FIFO. (ALways 0 when read.)

PCI Interrupt and MAC 8100 Configuration Register

Size 32-bit

I/O read-write

Address 0xC4

Bit	EDT_	Description
D31–16		not used
D15	PCI_EN_INTR	Enable PCI interrupt.
D14	RMT_EN_INTR	Enable interrupt from the MAC 8100.
D13	RX_MSG_EN_INTR	Enable receive message data available interrupt. A value of 1 enables the interrupt when bit 13 of the Fiber Message Interface register is set and bit 15 of this register is set.
D12	TX_MSG_EN_INTR	Enable transmit message register file empty interrupt. A value of 1 enables the interrupt when bit 31 of the Fiber Message Interface register is set and bit 15 of this register is set.
D10–11		not used
D9	MAC_UNRESET	A value of 0 resets the MAC 8100. A value of 1 starts normal operation.
D8	RFIFO_ENB	Enable DMA read FIFO
D 7–0	RMT_ADDR	MAC 8100 register address

PCI Interrupt Status Register

Size 32-bit

I/O read-only

Address 0xC8

Comments The driver uses this register initially to determine the source of a PCI interrupt.

Bit	PCD_	Description
D16–31		not used
D15	PCI_INTR	PCI interrupt. When asserted, the PCI FOI is asserting an interrupt on the PCI bus.
D14		not used
D13	RMT_INTR	Remote MAC 8100 interrupt. When asserted, the remote MAC 8100 interrupt is set. If bits 14 and 15 of the the PCI interrupt and MAC 8100 Configuration register are asserted, the MAC 8100 causes a PCI interrupt.

Bit	PCD_	Description
D12	DMA_INTR	End of DMA interrupt. Asserted when at least one of the DMA interrupts is asserted in the scatter-gather DMA next count and control register. Causes a PCI interrupt if bit 15 of the PCI interrupt and MAC 8100 Configuration register is set.
D11	RX_MSG_INTR	Receive message data available interrupt. Reflects the state of bit 13 of the PCI interrupt and MAC 8100 Configuration register. If bits 13 and 15 of the the PCI interrupt and MAC 8100 Configuration register are asserted, the MAC 8100 causes a PCI interrupt.
D10	TX_MSG_INTR	Receive message data available interrupt. Reflects the state of bit 12 of the PCI interrupt and MAC 8100 Configuration register.If bits 12 and 15 of the the PCI interrupt and MAC 8100 Configuration register are asserted, the MAC 8100 causes a PCI interrupt.
D 9–0		not used

MAC 8100 Data Register

Size 32-bit

I/O read-write

Address 0xCC

Bit	Description
D31–16	not used
D15–0	Read or write data for MAC 8100, using the address specified in EDT_RMT_ADDR (bits 0–7) of the main DMA command and configuration register.

Registers on the MAC 8100

The registers on the MAC 8100 (the SEEQ 8100 Gigabit Ethernet Media Access Controller) are initialized as follows. For detailed documentation on the MAC 8100 registers, see http://www.seeq.com.

Register Number	Register Name	Initialized Value (Hexadecimal)
7	Configuration 1	07C2
8	Configuration 2	0EC0
9	Configuration 3	2400
17	Transmit FIFO threshold	843F
18	Receive FIFO threshold	4080
19	Flow Control 1	4000
21	AutoNegotiation Advertisement	All zeroes

Registers not listed in this table are left in their reset state.

Specifications

PCI Bus Compliance		
Number of slots	1	
Transfer size	Maximum 64 bytes per transfer	
DVMA master	Yes	
PCI Bus memory	Approximately 66 KB	
Clock rate	33 MHz	
Device Data Transfer		
Protocol	Synchronous stream	
Buffers	4 KB FIFO for input, 2 KB FIFO for output	
Software	Drivers for Solaris 2.6+ (Intel and SPARC platforms), Windows NT Version 4.0, AIX Version 4.3, IRex 6.5, and Linux Red Hat Version 5.1	
Transceiver options	1.25 gigabaud 1300 ηm laser transceiver for longer distances 1.25 gigabaud 850 ηm VCSEL transceiver for lower cost	
Fiber Optic Cable		
1300 ηm laser transceiver (black shell)	*fiber type* 8/125 μm single mode fiber 62.5/125 μm multimode fiber	*distance* 3 km 550 m
850 ηm VCSEL transceiver (blue shell)	*fiber type* 62.5/125 μm multimode fiber	*distance* 300 m
Connectors	SC duplex fiber optic connector between fiber optic cable and transceive	
Laser safety	Class 1, eye-safe under a single fault condition	
Power	5 V at 2 A	
Environmental		
Temperature	Operating: 10 to 40° C Nonoperating: –20 to 60° C	
Humidity	Operating: 20 to 80% noncondensing at 40° C Nonoperating: 95% noncondensing at 40° C	
Physical		
Dimensions	3.3" x 5.05" x 0.5"	
Weight	3.3 oz.	

Table 6. PCI Bus Remote Camera Fiber Optic Interface Specifications

Index

P

R

S

T

U

V

W

X