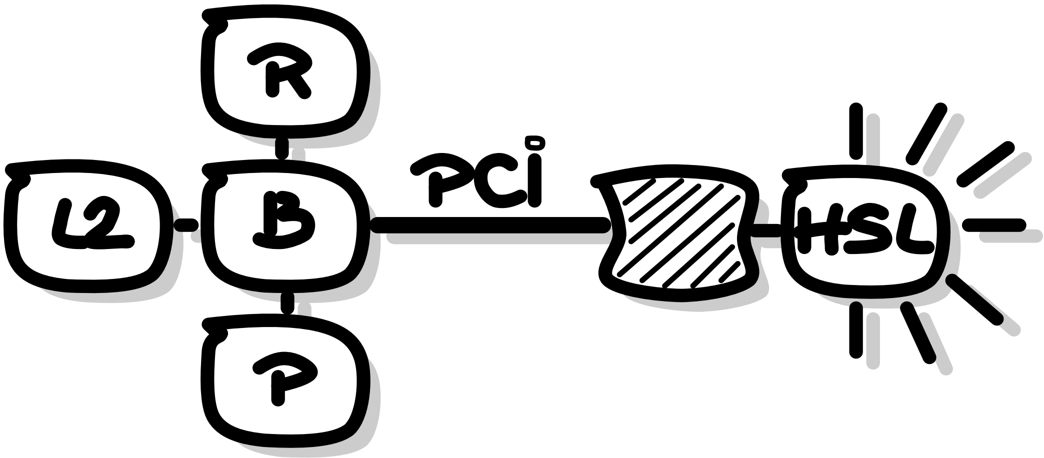
PCI-DDC

S P E C I F I C A T I O N S



Version 1.3a December 1996

Alain Greiner
Jean-Lou Desbarbieux Jean-Jacques Lecler Frédéric Potter Franck Wajsbürt
Stéphane Penain Cyril Spasevski

Content

CHAPTER 1

Document overview

This document contains the PCI-DDC component specification. That chip provides an interface between the RCUBE router and the PCI-local bus.

1.1 Scope

This specification describes how to use and program the PCI-DDC chip. It includes behavioural, electrical and physical description.

1.2 Contacts

Laboratoire MASI
Université Pierre et Marie CURIE
4 Place Jussieu
75252 Paris Cedex 5

1.3 Applicable documents

IEEE 1355	Heterogeneous InterConnect (HIC) Standard	D1.4, Feb 1995
PCI Special Interest Group	PCI Specification	Rev 2.1, Jun 1995
UPMC-MASI	RCUBE Specification	Ver 1.6, Feb 96

1.4 Abbreviations

DDC	Direct Deposit Component
DMA	Direct Memory Access
HSL	High Speed Link
LPE	List of Page to emit
LRM	List of Received Message
LMI	list of Message Identifier
MI	Message Identifier
PCI	Peripheral Component Interconnect
PL	Page Length
PLSA	Page Local Start physical Address
PRSA	Page Remote Start physical Address
RCUBE	Rapid Reconfigurable Router
RP	Routing Part

CHAPTER 2

PCI-DDC overview

This chapter gives a general presentation of PCI-DDC. Basic Terms are defined and general behaviour is described. Technical details can be found in chapter 3.

2.1 Function

PCI-DDC is a network interface component for message-passing multiprocessor architectures using HSL-network and the RCUBE router. It can be used with any CPU board having a PCI-local bus. PCI-DDC is a PCI bus master achieving Direct user Memory Access. It implements the Direct Deposit State Less Receiver Protocol (DDSLRP), developed at MASI, which reduces the processor overhead.

2.2 Definitions and Principles

2.2.1 PCI-bus

The PCI-bus is a standard I/O bus for personal computers and workstations. The targeted PCI-bus is PCI 2.1: 5-Volt / 32-bit / 33 MHz. The peak data throughput is then 132 MBytes/second. PCI-DDC acts as a master on the bus to receive and send data from and to the HSL-network.

2.2.2 HSL-network

The HSL-network is based on the IEEE.1355 High Speed Link (HSL) protocol used by the RCUBE router (Rapid Reconfigurable Router) designed and developed at the MASI laboratory. RCUBE is a low latency router (150 ns). It includes 8 bidirectional High Speed serial Link, a 8×8 non blocking crossbar switch and 8 independents programmables routing tables (one for each input link). Each link deliver a throughput of 1 Gbits/second (80 MBytes/second) in each direction. The architecture is fully parallel and achieves a peak data throughput of 640 MBytes/second.

RCUBE can cope with packets of any length. It implements a wormhole flow control technique, prefix and interval routing schemes. It provides efficient supports for adaptivity.

Port 0 and 1 of the RCUBE router can be directly accessed through a byte parallel interface. The PCI-DDC component interfaces directly the RCUBE router through this parallel interface (say port 0).

2.2.3 Direct Deposit StateLess Receiver Protocol (DDSLRP)

Classical data transfer protocols usually require several copies of data in intermediate buffers before and after transmission through the network.

In order to enhance performances, PCI-DDC implements a "remote write" primitive that directly fetch data from the local memory - in user space - and writes data directly into the remote memory - in user space too. This is a "Direct Deposit" Protocol. Furthermore, the sender processor defines the physical address where it wants the data to be written into the remote memory. This is a "StateLess Receiver" Protocol.

2.2.4 Memory address

All memory addresses handled by PCI-DDC are physical addresses. PCI-DDC provides no mechanism to translate virtual addresses to physical addresses, that is left to software.

2.2.5 Message

From the application software point of view, the transfer unit is the message. A message is usually a continuous space in the virtual memory. Message length can be anything from 1 byte to several Megabytes. The messages are not necessarily word-aligned. A message is transmitted from the local user memory to the remote user memory. There are several message types, depending on the synchronization mechanism.

2.2.6 Page

Since a message is in the virtual memory, it is usually fragmented in physical memory, and therefore must be split in pages before transmission. From PCI-DDC point of view, a page is a continuous space in the physical local memory, which contains data that have to be written in a continuous space into the physical remote memory. Therefore, a message is a list of one or several pages.

2.2.7 List of Pages to Emit (LPE)

PCI-DDC uses a List of Pages to Emit (LPE), implemented in the main memory of the sender, to send messages. All pages corresponding to a given message must be written that list by the software: the split of a message into physical pages must be done by software. All page descriptors for a given message must have consecutive entries in the LPE.

2.2.8 Page descriptor

The LPE contains page descriptors, not pages. Each one contains a Message Identifier (MI), a destination node number called Routing Part (RP), a local physical address (PLSA) which is the start address of the page in the local memory, a Page Length (PL) in bytes and a remote physical address (PRSA) which is the start address for the page in the remote memory. There are also few bits to code the message type, and protocol options.

2.2.9 Message Identifier (MI)

Each message is identified by a number called Message Identifier (MI), usually allocated by the operating system. All pages associated to the same message must have the same MI. The MI is used as an index by PCI-DDC to reconstruct messages in the remote memory and detect message completion. For that reason, PCI-DDC should not receive two messages at the same time with the same MI. The MI allocation is done by software and must respect the unicity principle for all nodes in the network. However, few types of message have not this constraints (see page 7).

2.2.10 Packet

Pages are the smallest entities from the software point of view but not from the hardware point of view. Pages are broken into packets by PCI-DDC. The packet is the physical exchange unit. The packet length is variable, and is dynamically adapted to the PCI burst length and the HSL-network. PCI-DDC does not guarantee that a page will be transferred as one packet if page length is more than one aligned word. Each packet contains all the informations necessary to reconstruct the message in the remote memory (including the physical remote address and the message identifier).

Basically, PCI-DDC try to send a page in only one packet, but a page will be split in the following cases:

- There is contention on the HSL-network. It is no worth locking the PCI-bus and PCI-DDC closes the packet.
- PCI-bus is preempted (due to latency timer mechanism for example). It is no worth locking HSL-network and PCI-DDC closes the packet.
- The Maximum Packet Length (MPL) has been reached, and PCI-DDC closes the packet. This maximum length is software defined by a configuration register (see chapter 3).

2.2.11 Message reconstruction

All packets corresponding to a given message are directly written in the destination processor memory by PCI-DDC. PCI-DDC must detect the completion of a given message and notify the destination processor. According to the routing tables used in the RCUBE routers, the HSL-network can be adaptive or not. When the network is adaptive, packets can follow several paths from a given sender to a given receiver. In that case the received packet order can be different from the sent packet order. When the network is not adaptive, there is only one path between any two nodes. Therefore, the received packet order is the same as the sent packet order. Adaptivity is an advantage from hot spot and fault tolerance point of view but the arrival order is not guaranteed. There is two possible methods to detect message transfer completion: This is defined in a configuration register (see chapter 3).

- To detect the completion of a message transfer in adaptive networks, PCI-DDC must use a List of Received Messages (LRM) implemented in main memory. The LRM is used to keep the current packet number for each message currently being received. PCI-DDC uses the MI as an index in the LRM. The last packet sent of a message contains the number of packet sent which is the expected packet number from the receiver point of view. Therefore, a message is completed when the received packet number is equal to the expected packet number. The LRM use is mandatory in adaptive networks. The LRM is useful in non-adaptive networks, because lost packets are always detected. The LRM is only managed by PCI-DDC, and thus do not imply processor overhead.

- For non-adaptive networks, it is possible to detect message completion without using the LRM. This can save memory (the LRM is a large table in physical memory) and decrease the traffic on the PCI bus. In this case, message completion is detected by the Last Packet bit contained in the last packet of a message. With this method, lost packets are not detected.

2.2.12 FIFO queues

PCI-DDC embeds FIFO queues (one per direction) in order to cope with the clock speed difference between the RCUBE router and PCI-bus.

When PCI-DDC emits data from the PCI-bus to RCUBE, until transmitting FIFO queue is not full, PCI-DDC can absorb the speed difference. Whenever the PCI-bus data throughput is lower than the RCUBE data throughput, the queue is not useful, since it cannot be filled. But whenever the PCI-bus data throughput is bigger than the RCUBE data throughput, the FIFO queue size defines the packet length, since the packet ends as soon as the queues is filled. The bigger the FIFO queue, the better throughput difference will be absorb. The transmitting FIFO queue size is **200** bytes.

When PCI-DDC receives data from RCUBE, receiving FIFO queue allows PCI-DDC to quickly flush the network, and to have enough time to ask for the bus. Furthermore, PCI-DDC can receive several messages from several senders at the same time, and so the receiving FIFO queue must be bigger than the transmitting one. The receiving FIFO queue size is **400** bytes.

Actually, those queue sizes could be discussed. They depend on lots of parameters such as message size, page size, time to obtain the PCI-bus, and so on. However, thoses sizes are a good tradeoff between silicon area and speed.

2.3 PCI-DDC notification

PCI-DDC can notify the receiving processor whenever a message has been completely received. The basic mechanism is to write the message identifier in a structure in the main memory (the List of Message Identifier: LMI), and to assert an interrupt signal. Thanks to the LMI, there is no contention at the receiving site: the software can handle several received message at a time.

PCI-DDC can notify the sending processor whenever a page has been completely transmitted by asserting an interrupt signal. Those interruptions are non-blocking from PCI-DDC point of view.

2.4 The remote write primitive

2.4.1 Principle

From the hardware point of view, a message transmission procedure consists of three stages: preparation, transmission and reception.

- **Preparation**

  Page Descriptors of the message are pushed by software into the LPE. (See page 19 for more details). Then, the software informs the PCI-DDC component of the added page number by writing in a PCI-DDC configuration register. Software must lock those pages during the transmission.

- **Transmission**

  PCI-DDC, at the sender side, asks for the PCI-bus and reads the LPE using DMA accesses in master mode. Then, it performs data transmission, using DMA accesses. There is no overhead for the processor during the transmission. The physical remote address is transmitted with every packet. When the last page is sent, PCI-DDC can notify the sending processor with an interrupt signal (this can be defined on a message base).

- **Reception**

  At the receiver side, as soon as PCI-DDC begins to receive packets, it asks for the PCI-bus and writes data at the specified address into the main memory using DMA accesses in master mode. There is no overhead for the processor during the reception. When the last page is written, PCI-DDC can notify the processor, either with an interrupt signal, and/or by writing in the LMI.

2.4.2 Message types

There are three types of messages : standard messages, short messages, composed messages.

- **Standard message**

  There is one transmitter and one receiver. The message consists of one or several pages described in the LPE. Data are fetched and written in user memory space.

- **Short message**

  A short messages contains 8 bytes or less. Data are fetched directly in the LPE and written directly in the LMI. The protocol is simpler and several PCI bus accesses are saved.

- **Composed message**

  There are several transmitters and one receiver. The receiver waits until all transmitters have sent one empty message. This message type which does not transfer any data, can be used to implement synchronization. barriers. All piece of message of a composed message must have the same Message Identifier (MI).

2.5 Error detection

No automatic error recovery is incorporated in PCI-DDC. Nevertheless, there is several error detection mechanisms detailed in the next chapter (page 13):

Network errors . . whenever packet has been corrupted in the network ;

PCI-bus errors . . . whenever parity error has been detected or bus transaction has aborted ;

Internal errors . . . whenever internal counters or memory structures has overflowed ;

RCUBE errors . . . whenever PCI-DDC input pins reporting the RCUBE status has gone up.

In all cases except for the RCUBE errors, PCI-DDC stops what it is doing, notifies the processor by an interrupt and waits for an acknowledge. For the RCUBE errors, PCI-DDC just asserts an interrupt.

The error detection avoids to write corrupted data in remote memory and prevents from a local or remote system crash. The software must decide if an error is recoverable or if a global reset is needed. Every error may be hidden with a mask bit, whenever the software does not want to be informed. However, whenever masked errors occur, they are detected and reported in the error register. The difference concerns the interrupt line assertion.

CHAPTER 3
PCI-DDC programming

This chapter describes how to use and to program PCI-DDC. It begins with a description of all the visible registers that must be handled by the PCI-DDC driver, describes the memory data structures, the data exchange formats, and finally the PCI-DDC starting procedure.

3.1 Configuration registers

There are two configuration spaces for PCI device. The first one consists of 16 standard 32 bits registers and the second one is specific for each device. The first part is succinctly described, the most important point being the default value. It is strongly advised to read the PCI-bus specification to have more informations.

3.1.1 Register access modes

Registers are 32-bit wide but maybe split in fields. Each field has an independent access mode:

(R) . . . . Register may be read, and write a value does not matter.
(W) . . . . Register may be written and, a read value is undefined.
(R/W) . . . . Register may be read and written.
(R/C) . . . . Register may be read and cleared.
Write a 0 does not change the field, write a 1 clears it.

Access to an undefined register, read a **W** type register or write a **R** type register does not matter. In that cases, write has no effect and read give an undefined value back. In the following description, *set a bit* means write 1, *reset a bit* means writes 0, moreover *test if a bit is on* means compare at 1.

3.1.2 Register map

This map describes the register space. The first one concerns the standard registers. Few registers are unused since PCI-DDC contains neither RAM nor ROM nor even PCMCIA device. Thus, only the arrow right tagged registers are used and therefore described below.

	31 … 24	23 … 16	15 … 8	7 … 0	
00 h	Device ID		Vendor ID		p. 9 ◁-
04 h	Status		Command		p. 10 ◁-
08 h	Base Class Code	Sub Class Code	Programming Interface	Revision ID	p. 12 ◁-
0C h	BIST	Header Type	Latency Timer	Cache Line Size	p. 12 ◁-
10 h – 24 h	Base Address Register				
28 h	Cardbus CIS Pointer				
2C h	Subsystem ID		Subsystem Vendor ID		
30 h	Expansion ROM Base Address				
34 h	Reserved				
38 h	Reserved				
3C h	Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	p. 12 ◁-

Figure 3.1: Type 00h configuration header space

The second map presents the PCI-DDC specific registers:

	31 … 8	7 … 0	
40 h	PCI-DDC Status	R3 Status	p. 13
44 h	PCI-DDC Status Mask	R3 Status Mask	p. 16
48 h	PCI-DDC Command		p. 17
4C h	Message Type (31–24) / Received Message Identifier (23–0)		p. 18
50 h	LPE Maximum Address		p. 19
54 h	LPE Current Address		p. 19
58 h	LPE New Address		p. 19
5C h	LPE Base Address		p. 19
60 h	LMI Maximum Address		p. 20
64 h	LMI Current Address		p. 20
68 h	LMI New Address		p. 20
6C h	LMI Base Address		p. 20
70 h	LRM Base Address		p. 22
74 h	Reserved	R3 Dout	p. 23

Figure 3.2: PCI-DDC configuration space

3.1.3 Device and Vendor Identification

	31 … 16	15 … 0
	(R) Device ID	**(R) Vendor ID**
00 h	FFFFh	xxxxh

Figure 3.3: Device and Vendor Identification

Vendor Identifier is assigned by PCI SIG. (NOT DONE YET !)

3.1.4 Standard Status and Command Register

Since PCI-DDC does not use all functionalities, most command bits are just readable.

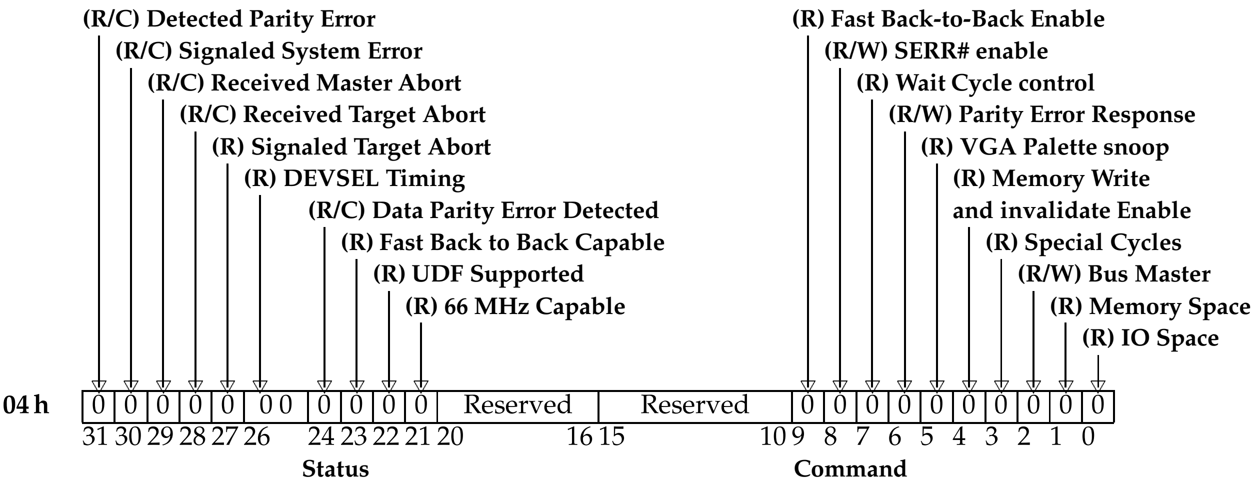


Figure 3.4: Status and Command

<u>Status</u>

Detected Parity Error . . . . A data or address parity error on the PCI bus has been detected by PCI-DDC. This bit is set only when PCI-DDC has to check the parity gives by another agent. This bit is not maskable, however it never brings interruption.

Signaled System Error . . . . PCI-DDC sets this bit when it has detected an address parity error and if the two bits SERR enable and Parity Error Response (PCI command register) are on. When this bit is set, PCI-DDC also asserts the INTA# signal and resets the Bus Master bit (PCI command register).

Received Master Abort . . . . When PCI-DDC is acting as a master on the PCI bus, it puts an address on the bus. If no memory decodes the specified address as its own or if no physical memory is wired at that address, then the transaction ended with a Master-Abort. In that case, PCI-DDC sets the Received Master Abort bit, asserts the INTA# signal and resets the Bus Master bit (PCI command register).

Received Target Abort . . . . A target agent has aborted an access. It means a major problem has happened. In that case, PCI-DDC sets the Received Target Abort bit, asserts the INTA# signal and resets the Bus Master bit (PCI command register).

Signaled Target Abort . . . . PCI-DDC never aborts a transaction by itself. This bit is stuck at 0.

DEVSEL Timing . . . . PCI-DDC is a fast type of device.

Data Parity Error Detected . . . . A data parity error on the PCI bus has been detected by PCI-DDC acting as a master. This bit is set only if Parity Error Response (PCI command register) bit is set. When this bit is set, PCI-DDC also asserts the INTA# signal and resets the Bus Master bit (PCI command register).

Fast Back-to-Back Capable . . . . PCI-DDC is not able to accept fast back-to-back transaction. This bit is stuck at 0.

UDF Supported . . . . PCI-DDC does not support software selectable configuration items. This bit is stuck at 0.

66Mhz . . . . This is a 33 Mhz version. This bit is stuck at 0.

Command

Fast Back-to-Back Enable . . . . PCI-DDC is not able to do fast back-to-back transactions. This bit is stuck at 0.

SERR# Enable . . . . When this bit is set and also when the Parity Error Response bit is set, PCI-DDC can report address parity error on SERR#.

Wait Cycle Control . . . . PCI-DDC does not use Address or Data Stepping. This bit is stuck at 0.

Parity Error Response . . . . When this bit is set, PCI-DDC checks parity.

VGA Palette snoop . . . . The interface is not a display peripheral. This bit is stuck at 0.

Memory Write and Invalidate . . . . . . PCI-DDC does not use this PCI command. This bit is stuck at 0.

Special Cycle . . . . PCI-DDC does not respond to PCI special cycles. This bit is stuck at 0.

Bus Master . . . . This bit Allows PCI-DDC to act as a master on the PCI Bus. When this bit is 0, PCI-DDC cannot transmit or receive data to or from the network. This bit is cleared by the reset signal or if any error is detected by PCI-DDC, (except for R3 Status Error, see PCI-DDC status register). As soon as this bit is set, PCI-DDC asks for the PCI-bus, whenever something must be sent or received. So, be sure to have a good configuration register state before setting it (see PCI-DDC initialization page 28 for more details).

Memory Space . . . . PCI-DDC does not answer to memory space accesses. This bit is stuck at 0.

IO Space . . . . PCI-DDC does not answer to I/O Space accesses. This bit is stuck at 0.

3.1.5 Class Code

	(R) Class	(R) SubClass	(R) Programming Interface	(R) Revision ID
08 h	02h	80h	00h	01h
	31 … 24	23 … 16	15 … 8	7 … 0

Figure 3.5: Device Class Code

PCI-DDC is a network controller. This field contains 02h.

Sub Class .... PCI-DDC has not a standard class. That is another network controller so this field contains 80 h.

Programming Interface .... There are no register-level programming interfaces. This register is stuck at 00 h.

Revision ID .... That is the first revision.

3.1.6 Miscellaneous Functions

BIST .... There is no built-in self test in PCI-DDC.

Header Type .... PCI-DDC is a single function device.

Latency Timer .... The maximum number of cycles a PCI-DDC access bus can last, when somebody else requests the bus. If the latency timer is 00h, the transaction lasts while the GNT# signal is asserted.

Cache Line Size .... PCI-DDC is not capable of Memory-Write and Invalidate accesses. This field is stuck at 0.

	(R) BIST	(R) Header Type	(R/W) Latency Timer	(R) Cache Line Size
0C h	00h	00h	00h	00h
	31 … 24	23 … 16	15 … 8	7 … 0

Figure 3.6: Device Miscellaneous Functions

3.1.7 Device Requirement

Maximum Latency .... The transfer throughput regularity is not a critical point, the required maximum latency is 0 micro-second.

Minimum Grant..................... The transfer throughput regularity is not a critical point, the required minimum grant is 0 micro-second.

Interrupt Pin .... PCI-DDC is a single function peripheral: interrupt pin is therefore INTA.

Interrupt Line .... This 8-bit R/W register is reserved for software use.

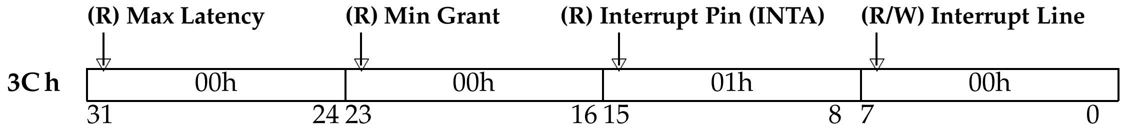


Figure 3.7: Device Requirement

3.1.8 PCI-DDC Status Register

This register reports the normal events (ie: Message Received bit and Page Transmitted bit) and error events, it is the PCI-DDC status. There is a mask bit for each status bit. If a mask is 1, as soon as the associated event happens then the INTA# signal is asserted, else the mask is 0 then the event is ignored. If it is an error event (except for R3 Status Error) then the Bus Master bit (PCI command register) is reset and PCI-DDC waits for a configuration transaction (for example to read the PCI-DDC Status Register and restarts PCI-DDC). PCI-DDC does not stop after a normal event or R3 Status Error. In the following description, we suppose that every event can be hidden by its associated mask.

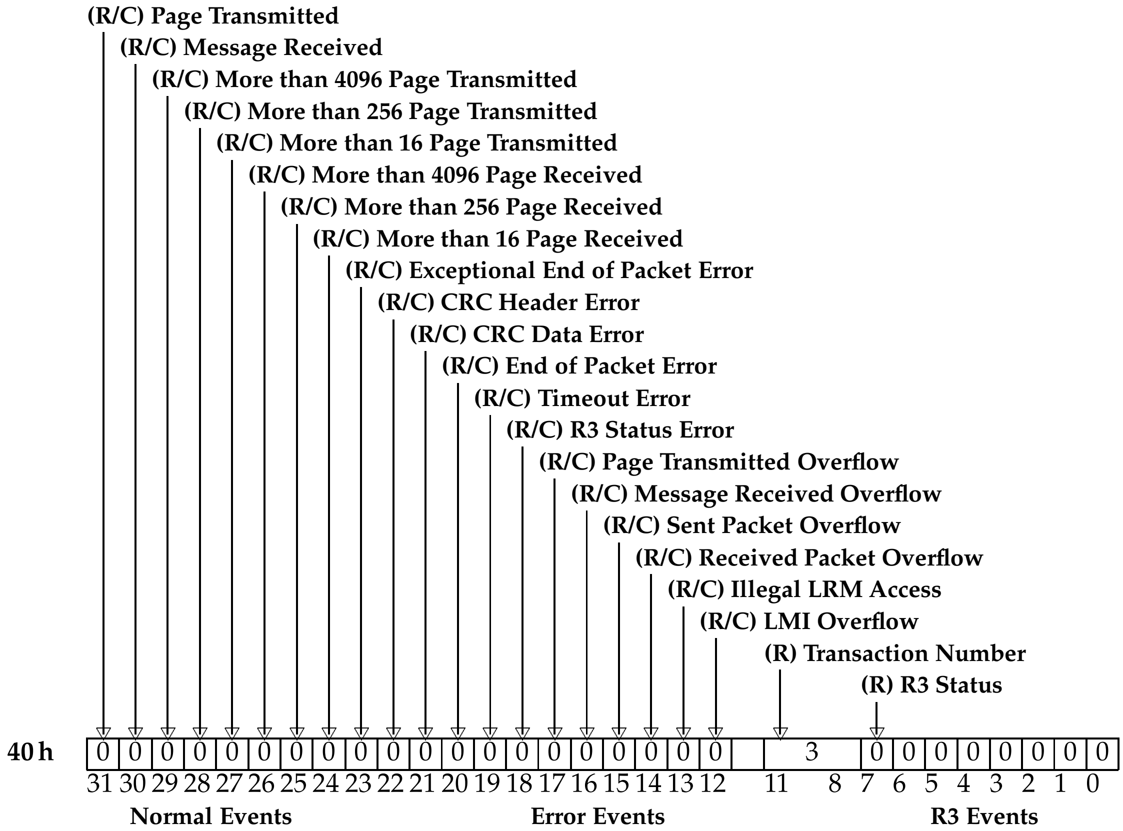


Figure 3.8: PCI-DDC Status

PCI-DDC normal events

Page Transmitted . . . . This bit is set whenever a full page has been transmitted and if the software has set the Notify Once Sent (NOS) bit for this page (see Page Descriptor format page 23 for more details). This event does not stop PCI-DDC. If PCI-DDC sends another page with a NOS bit before the processor has acknowledged the previous page, the Page Transmitted bit is set again. To reset that bit, the processor has to clear it as many times as it has been set by PCI-DDC (clear means write one in it). Actually, There is a counter for the transmitted page with NOS bit and this bit reports this counter value, so one means greater than 0. The maximum counter value is $2^{20} - 1$, which imply a Page Transmitted Overflow detection. (see below).

Message Received .... This bit is set whenever a full message has been received and if the software has set the Notified Once Received (NOR) bit for this message (see Page Descriptor format page 23 for more details). This event does not stop PCI-DDC (see LMI description page 20). If PCI-DDC received another message with a NOR bit before the processor has acknowledged the previous message, the Message Received bit is set again. To reset that bit, the processor has to clear it as many times as it has been set by PCI-DDC (clear means write one in it). Actually, There is a counter for the received message with NOR bit and this bit reports this counter value, so one means greater than 0. The maximum counter value is $2^{20} - 1$, which imply a Message Received Overflow detection (see below).

Intermediate filling level of Page Transmitted

PCI-DDC allows the software to know the filling level of transmitted page counter. There are three bits dedicated to that function. The first one (bit 29) indicates if there are more than 4096 pending transmitted page. Clear that bit means clear 4096 interrupt at one time. Clear a zero does not do anything. The second (bit 28) and third (bit 27) have the same behaviour but with different levels.

Intermediate filling level of Message Received

PCI-DDC allows the software to know the filling level of received message counter. There are three bits dedicated to that function. The first one (bit 26) indicates if there are more than 4096 pending received message. Clear that bit means clear 4096 interrupt at one time. Clear a zero does not do anything. The second (bit 25) and third (bit 24) have the same behaviour but with different levels.

PCI-DDC error event

Exceptional End of Packet Error .... is set if a received packet ends with the EEP character. This EEP character is sent by a router in order to report an error in the HSL-network, such as a calibration loss or a parity error (See RCUBE specification for more details).

CRC Header Error .... is set if the CRC on a received packet header is wrong. In that case, the corrupted packet is not written in memory. (see Packet Format page 26 for more details).

CRC Data Error .... is set if the CRC on a received packet data is wrong. When such an error is detected, data may have already been written into the memory (see Packet Format page 26 for more details).

End of Packet Error .... is set if the received packet has a wrong termination. A standard packet always ends with two control characters: an End of Data (ED) and an End of Packet (EP) (see Packet Format page 26 for more details). There is an error, if the EP character does not occur exactly 8 bytes after the ED character, or if EP occurs alone, or if an unexpected character Begin Packet (BP) is detected. When this error is detected, data may have already been written into the memory.

Timeout Error .... is set in case of abnormal condition: Actually, no action in PCI-DDC lasts more than few thousands of cycles. A timeout condition may happen if, for instance, a Begin Packet (BP) has been received but the End of Packet (EP) is not received. The timeout value is 2**22 cycles (about 1 second according to the PCI-bus clock frequency).

R3 Status Error .... is set if at least one of the 8 bits of the R3 Status register is one. That bit allows the software to ask for an interrupt when the R3 Status Register goes up.

Page Transmitted Overflow .... Each time PCI-DDC transmits a page with a NOS bit, a counter is incremented and the INTA# signal is asserted. If this counter is $2^20 - 1$, there is an overflow error. This error means there is there are $2^20 - 1$ pending interrupt requests.

Message Received Overflow .... Each time PCI-DDC receives a message with a NOR bit, a counter is incremented and the INTA# signal is asserted. If this counter is $2^20 - 1$, there is an overflow error. This error means there is there are $2^20 - 1$ pending interrupt requests.

Sent Packet Overflow .... PCI-DDC splits page into packets. A page can be split in 64 K packets at most. In case of counter overflow, this bit is set.

Received Packet Overflow .... The last packet of a message contains the total number of packets that have been sent for a given message. If more than 64 K packets are received, this bit is set.

Illegal LRM Access .... A message has been received which asks to write into the LRM, but the LRM Enable bit (PCI-DDC Command register) is currently disabled.

LMI Overflow .... is set if the LMI structure is full. It means that software has to free space and to move current pointer (see LMI format page 20). The LMI Overflow bit is set as soon as PCI-DDC writes the last element in the LMI.

internal status

PCI-DDC remembers the transaction iwhich has just been done before the current read configuration. In case of error, it allows the software to know which transaction is responsible of it.

All possible PCI transaction has got a number except configuation read or write.

LPE . . . . `1` : a page descriptor has been read.

TX . . . . `2` : one paquet has been sent.

RX . . . . `4` : one paquet has been received.

LRMW . . . . `5` : The LRM structure has just been written.

LRMR . . . . `6` : The LRM structure has just been read.

LMI . . . . `7` : The LMI message descriptor has just been written.

RESET . . . . `3` : No performed transaction since previous (hard/soft) reset.

RCUBE status

R3 Status . . . . When PCI-DDC is connected with a RCUBE router, this 8-bit register allows to report the calibration status of each HSL link (from the bit 0 for link 0 to the bit 7 for link 7). If a bit is one, the associated link has loss the calibration. Actually, this register might be used to read any 8 bits status. As soon as an unmasked bit of the R3 status is one, the R3 status Error bit is set.

3.1.9 PCI-DDC Status Mask Register

This register has the same structure as the PCI-DDC Status Register. It allows the software to hide independently events or errors.

PCI-DDC normal event status mask

Page Transmitted Mask . . . . When this bit is set, it enables the Page Transmitted interrupt. However, to have an interrupt asserted, the page must have the NOS bit (see Pages Descriptor page 23).

Message Received Mask . . . . When this bit is set, it enables the Message Received interrupt. However, to have an interrupt asserted, the message must have the NOR bit (see Pages Descriptor page 23).

PCI-DDC error event status mask

Exceptional End of Packet Error Mask
Should be 1. When 0, the Exceptional End of Packet Error is ignored.

CRC Header Error Mask . . . . Should be 1. When 0, the CRC Header Error is ignored.

CRC Data Error Mask . . . . Should be 1. When 0, the CRC data Error is ignored.

End of Packet Error Mask . . . . Should be 1. When 0, the End of Packet Error is ignored.

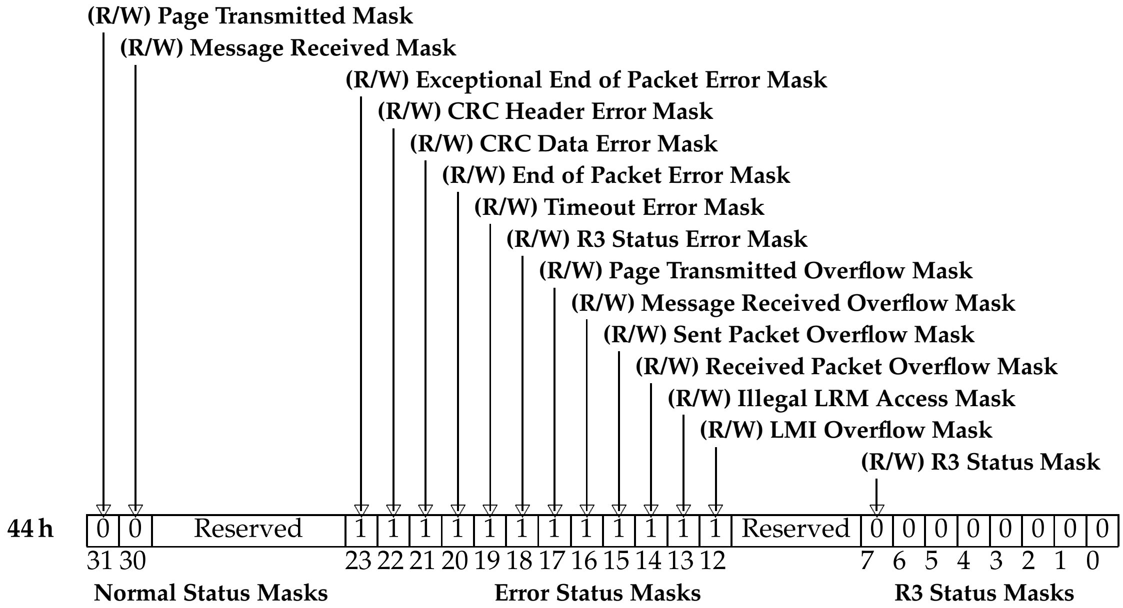


Figure 3.9: PCI-DDC Status

Timeout Error Mask .... Should be 1. When 0, the Timeout Error is ignored.

R3 Status Error Mask .... Should be 1. When 0, the R3 Status Error is ignored.

Page Transmitted Overflow Mask .. Should be 1. When 0, the Page Transmitted Overflow is ignored.

Message Received Overflow Mask . Should be 1. When 0, the Message Received Overflow is ignored.

Sent Packet Overflow Mask .... Should be 1. When 0, the Sent Packet Overflow is ignored.

Received Packet Overflow Mask ... Should be 1. When 0, the Received Packet Overflow is ignored.

Illegal LRM Access Mask .... Should be 1. When 0, the Illegal LRM Access is ignored.

LMI Overflow Mask .... Should be 1. When 0, the LMI Overflow is ignored.

RCUBE status mask

R3 Status Mask .... allows the software to mask each R3 Status bit. When a mask bit is 0, the associated status bit is hidden.

3.1.10 PCI-DDC Command

Transmit/Received Balance .... This 4 bits field allows the software to balance transmit priority versus receive priority. This field defines a threshold between 0 to 15. When PCI-DDC has to choose between reception and transmission, it uses a modulo 16 counter. If the current counter value is strictly greater than the software threshold, PCI-DDC starts a transmission, else it starts a reception.

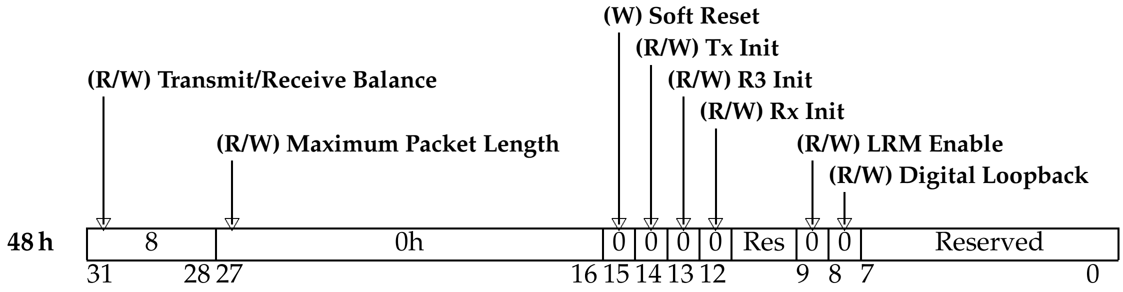


Figure 3.10: PCI-DDC Command Register

Maximum Packet Length .... This register (**MPL**) allows the software to limit the maximum packet length. The byte length (**L**) is defined by the equation: **L**=**MPL**$*$4 + 36. A zero value means the packets may have any length.

Soft Reset .... Set this bit has the same effect than assert the RST# signal. This bit is automatically reset after one cycle. All registers takes their default value, FIFO queues are flushed, the finite state machines returns to their idle state. PCI-DDC is then ready to be configured. (see PCI-DDC initialization page 28 for more details).

Tx Init .... Switch on the Transmission Block. (see PCI-DDC initialization page 28 for more details).

R3 Init .... Switch on the RCUBE component. (see PCI-DDC initialization page 28 for more details).

Rx Init .... Switch on the Reception Block. (see PCI-DDC initialization page 28 for more details).

LRM enable .... The LRM is an optional structure into the main memory that PCI-DDC uses to count received packets. This structure is mandatory when the network is adaptive, and is useful for a safe message completion detection. A 0 value means the LRM is not used: the message completion is detected by the Last Packet bit in the arriving packet. A 1 value means the LRM is used (and must be allocated in memory): The message completion is detected the received packet number is equal at the Expected Packet Number.

Digital Loopback .... Sets the loop back mode. It means that the output and input ports are strapped and all the packets sent are directly received. This is only for test purpose, in normal use, it should be kept at 0.

3.1.11 Received Message Identifier

This read-only register contains the Message Identifier of the last received packet and some useful flags. The flags are detailed page 23 with the page descriptor. For each new received packet, this register is modified. This register is written in the LMI when the message is completed. In case of error, this register is useful to debug.

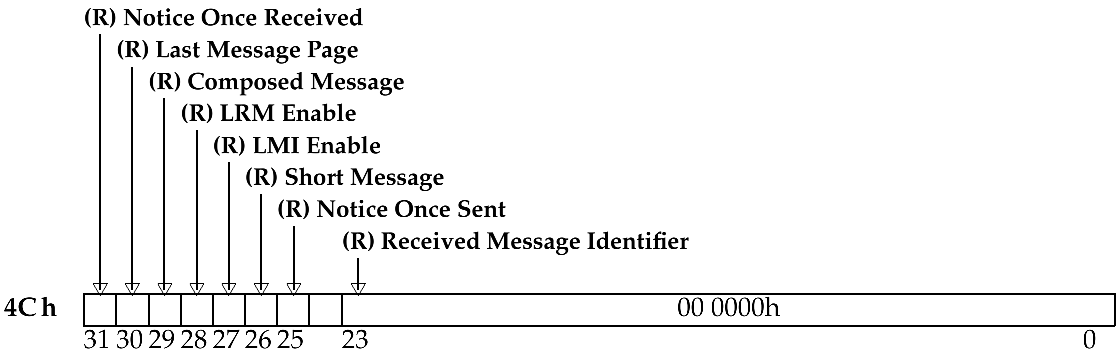


Figure 3.11: Received Message Identifier and Message Flags

3.1.12 LPE Management Registers

All remote-write request are placed by the PCI-DDC driver in a software queue, called the List of Pages to Emit (LPE). This list resides in main memory, it is written by the processor and it is read by PCI-DDC. It contains actually Page Descriptors presented page 23. The pages are transmitted in the queue order.

The LPE is a circular queue which needs four pointers : one to the Base address, one to the Maximum address, one to the Current transmitted page and one to the New page to transmit. All those pointers are physical addresses. Since a Page Descriptor is 16 bytes length, the four lowest bits are 0.

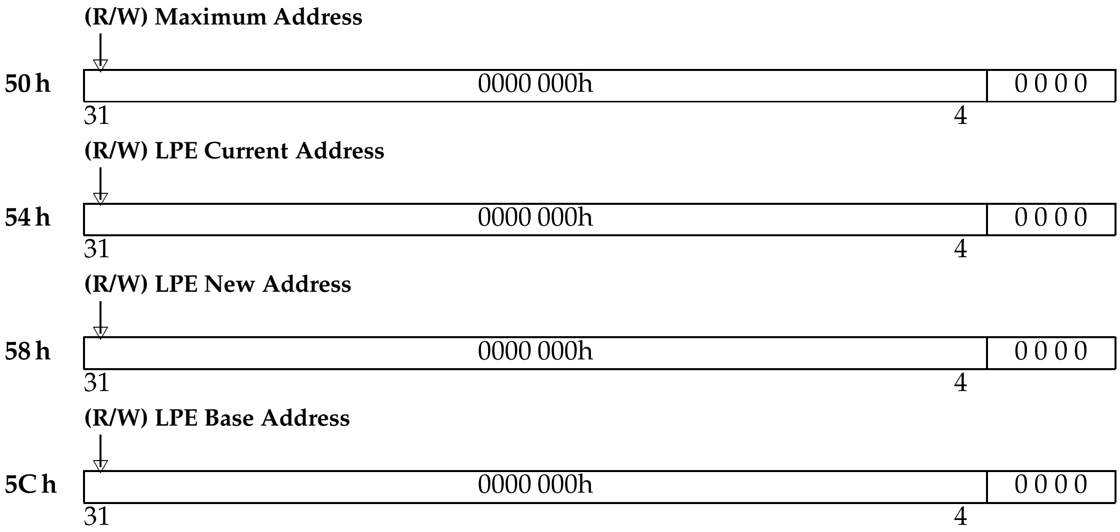


Figure 3.12: LPE Registers

LPE Maximum Address .... That is the highest Page Descriptor address. This register has to be set at the PCI-DDC initialization.

LPE Current Address .... That is the current address describing the page which is presently being transmitted by PCI-DDC Once a page has been sent, the LPE current address is incremented by PCI-DDC. If the LPE Current register was the LPE Top address then the next is LPE Base Address.

LPE New Address .... That is the last added page to be transmitted. This register is written by the software (using a configuration access on the PCI bus) each time a new page is added into the LPE.

LPE Base Address .... That is the lowest Page Descriptor address. This register has to be set at the PCI-DDC initialization.

At initialization phase, when those registers are set, the Bus Master bit (PCI command register) must be disasserted in order to avoid PCI-DDC to start a Page Descriptor read. When the PCI-DDC driver wants to add one or several Page Descriptors into the LPE, it must:

- read the LPE registers to check if there is enough space.
- Write the Page Descriptors in the LPE.
- Write the New Address in PCI-DDC.

The figure below highlights the address register usage.

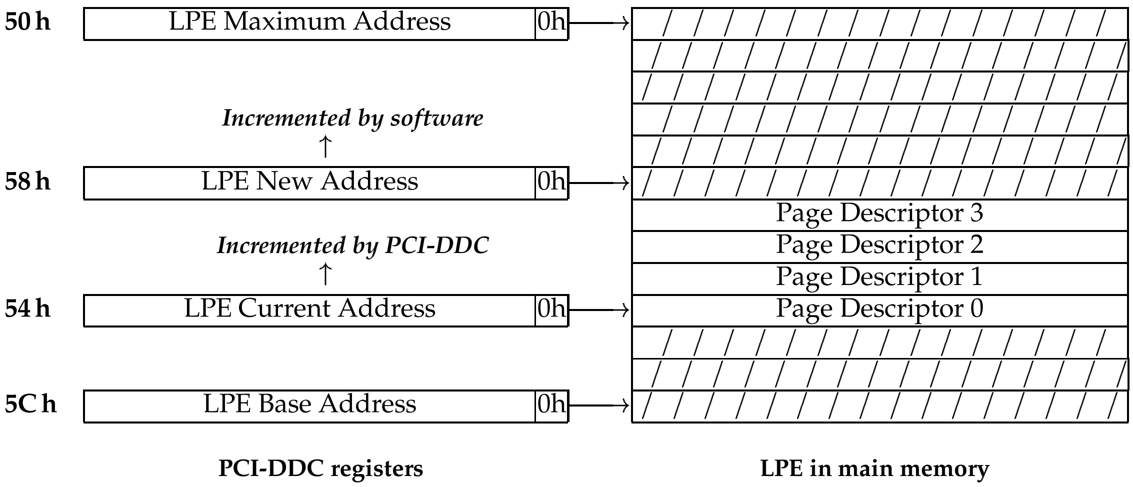


Figure 3.13: LPE management

The New Address has been incremented after the addition of the Page Descriptors 0 to 3. Then PCI-DDC knowns that it has to transmit pages beginning by 0 up to 3. When Current Address reaches the New Address, it means that PCI-DDC has no more page to transmit.

3.1.13 LMI Management Registers

Each received message having the LMI enable bit set (see Page Descriptors page 23) is reported in the List of Message Identifier. This software queue contains one Message Descriptor for each received message that is completed.

The LMI is a circular queue which needs four pointers : one to the Base address, one to the Maximum address, one pointer to the Current received message, and one to the New received message. All those pointers are physical addresses. Each element in the LMI is 16 bytes length, the four lowest bits are 0. The maximum is harwired to maximum memory address in order to not cause an overflow error just after reset event.

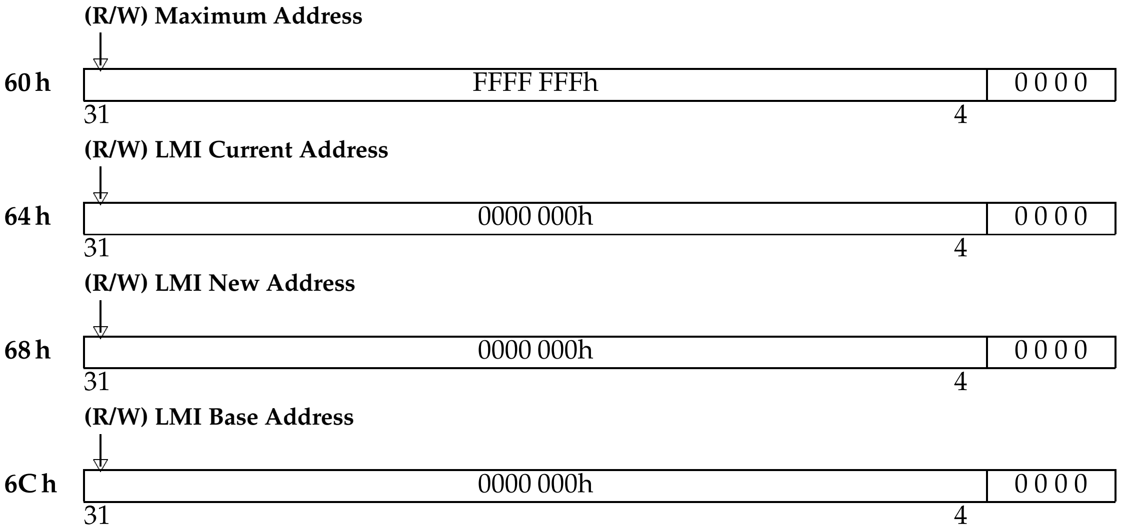


Figure 3.14: LMI Registers

LMI Maximum Address .... That is the highest Message Descriptor address. This register has to be set at the PCI-DDC initialization.

LMI Current Address .... That is the address of the first message that has to be consumed by the software. Once that message has been consumed, this register must be incremented by the software (using a configuration acces on the PCI bus)

LMI New Address .... That is the next free LMI entry where PCI-DDC will write the next Message Descriptor. This register is incremented by PCI-DDC. Then if LMI New Address equal to LMI Current Address, an error is detected.

LMI Base Address .... That is the lowest Message Descriptor address. This register has to be set at the PCI-DDC initialization.

At initialization phase, when those registers must to be set, the Bus Master bit (PCI command register) must be disasserted in order to avoid PCI-DDC start to receive a message from the network. The figure below highlights the address register usage.

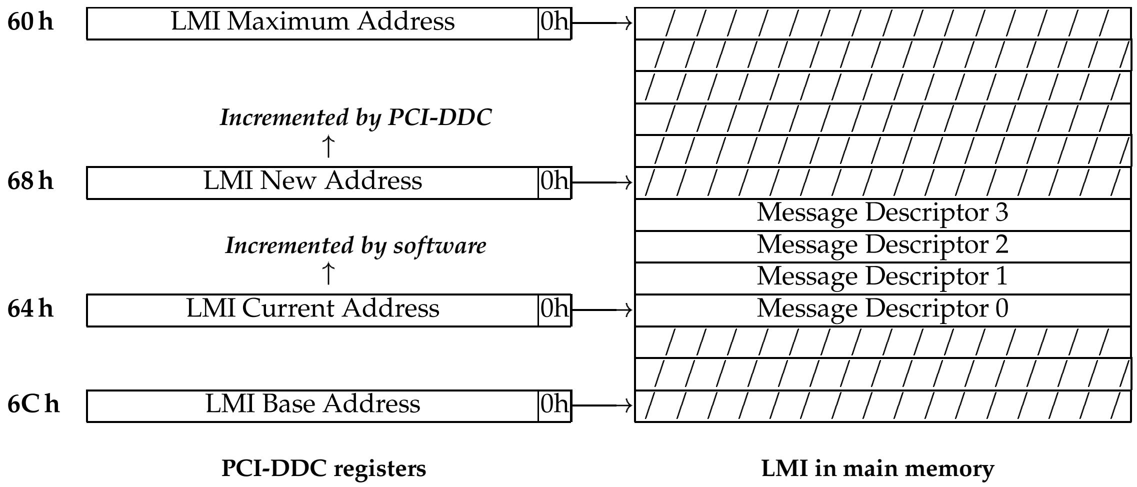


Figure 3.15: LMI management

The Current Address is incremented by the software once it has read the Message Descriptor 0 to 3. PCI-DDC will write the new Message Descriptor at LMI New Address. When, New Address reaches the Current Address, it means the LMI is full and PCI-DDC must wait for the software to free space.

3.1.14 LRM Base Address

When the network is adaptive, PCI-DDC must count the received packet for each message, using a specific structure in memory called List of Received Message. It contains the Received Message Descriptors presented page 25. The LRM Base Address is the lowest address of the list. The current Message Identifier is used as an index in the LRM. The actual size of the LRM is defined by the software. This defines a maximum value for the Message Identifier that must be respected by the software (there is no hardware verification of this maximal value).

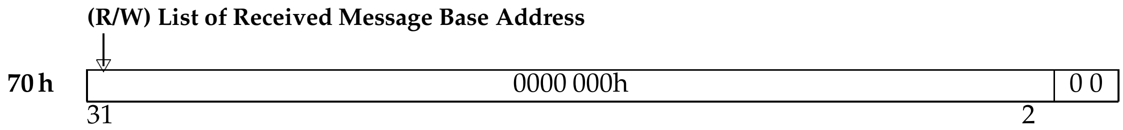


Figure 3.16: List of Received Message Base Address

Since the descriptor is 4 bytes length, the two lowest bits of the Base Address are 0 and the Message Identifier has to be multiply by 4 before to be added with the Base Address.

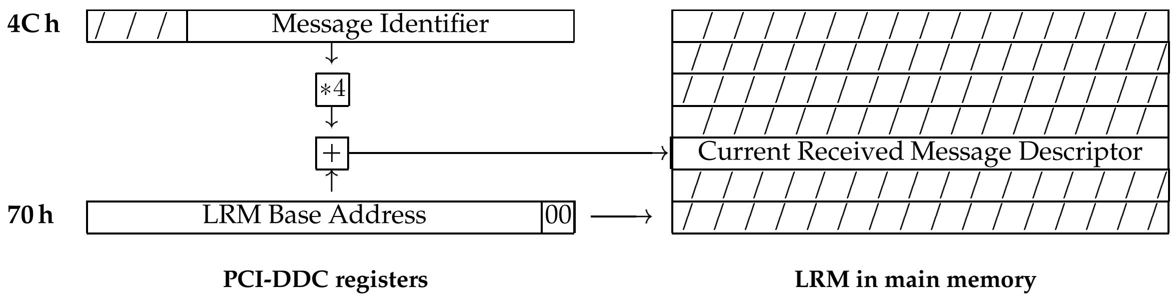


Figure 3.17: List of Received Message

3.1.15 RCUBE configuration

This register is designed to access the configuration port of the RCUBE router. This port is used to load the RCUBE routing tables. This initialization phase requires 258 write into the R3 Dout register. The R3 init bit has to be reset (maintained at 0) as long as the RCUBE is not configured. The complete RCUBE configuration space is detailed in the RCUBE specification. Actually, this port might be used to program any external component.

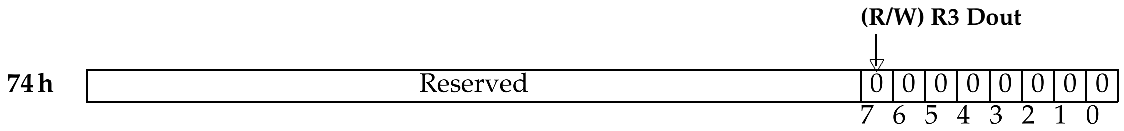


Figure 3.18: R3 Dout Register

3.2 Structures in main memory

PCI-DDC only uses three data structures in main memory. This section presents the Page Descriptor for the LPE, the Message Descriptor for the LMI and the Received Message Descriptor for the LRM.

3.2.1 Page Descriptor in the LPE

Each Page Descriptor is 16 bytes length consistings of the following fields:

Routing Part........................ Those 16 bits define the destination node number. It is the node address in the network. This is used by the RCUBE router.

Page Length........................ This is the number of bytes to be transferred. It should be lower than 65535 bytes. If the length is 0, the corresponding LPE entry is ignored by PCI-DDC.

Notice Once Received.............. When this bit is 1 in the last page of a message, an interrupt signal will be asserted at the receiver site when the message is completed. It may be invalidated by the Received Interrupt Enable bit in the receiver PCI-DDC.

Last Message Page .... When this bit is 1, the page is the last of the message. PCI-DDC transmits this information with the last packet of the page.

Composed Message .... When this bit is 1, the page is part of a composed message. That message does not contain data.

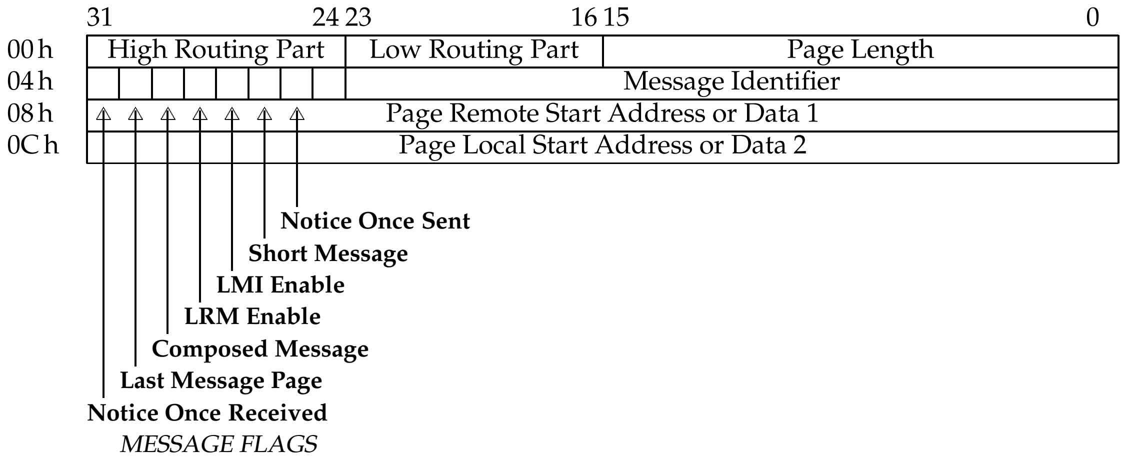


Figure 3.19: Page Descriptor

LRM Enable........................ When this bit is 1, PCI-DDC uses the LRM for message completion detection. When this bit is 0, PCI-DDC uses the Last Packet bit for message completion detection.(see Packet format page 26 and Message reconstruction page 5 for more details).

LMI enable......................... When this bit is 1, the receiver PCI-DDC writes the message identifier in the LMI, once the message is completed. When this bit is 0, there is no report in the LMI.

Short Message...................... When this bit is 1, the page descriptor contains a short message. It means the next two words are directly data to send, not pointers. The Page Length is ignored.

Notice Once Sent................... When this bit is 1, PCI-DDC asserts an interrupt signal once the page sent. That bit may be invalidated by the Transmit Interrupt Enable bit.

Message Identifier.................. Identifies the message. All pages belonging to the same message must have the same MI. If the LRM is used, the Message Identifier has a maximal value defined by software and corresponding to the LRM size.

Page Remote Start Address......... It is the physical address in the remote memory for the standard message, or the first data for the short message.

Page Local Start Address .... It is the physical address in the local memory for the standard message, or the second data for the short message.

3.2.2 Message Descriptor in the LMI

Each Message Descriptor is 16 bytes length consistings of the following fields:

R3 Status .... It is the R3 Status register value when the Message Descriptor has been written in the LMI. R3 Status register may change at each cycle since it reports R3STATUS pins, this information may be used for debug purpose.

Packet Number .... It is the final received Packet Number for the message.

Message flags .... Those are exactly the same flags as those present in the page descriptor, since those flags are put in the packet.

Message Identifier .... It is the message identifier.

Data 1 .... This field is used only when the current message is a short message. In that case, this is the first word.

Data 2 .... This field is used only when the current message is a short message. In that case, this is the second word.

	31 … 24	23 … 16	15 … 0
00 h	Reserved	R3 Status	Packet Number
04 h	Message Flags	Message Identifier	
08 h	Data 1		
0C h	Data 2		

Figure 3.20: Message Descriptor

3.2.3 Received Message Descriptor in the LRM

For a standard messages, the LRM is a private data structure in main memory for PCI-DDC. This structure should not be read or written by software but for composed message).

The total number of packets for a given message is computed by PCI-DDC and transmitted with the last packet of the message. The Received Packet Number is incremented by PCI-DDC at each packet received. A message is completed when the Received Packet Number matches the Expected Packet Number. There is one word dedicated for each MI allowing PCI-DDC to remember received and transmitted paquets after each packet received. All words are the LRM. In case of a Composed Message, the Expected Packet Number must be written by the software since he knows exactly the number of expected packets.

PCI-DDC suposes that structure empty (fill with 0) after reset. PCI-DDC deletes itself (written 0) word once a message is completely received.

	Expected Packet Number	**Received Packet Number**
MI	0000h	0000h
	31 … 16	15 … 0

Figure 3.21: Received Message Descriptor

Each Received Message Descriptor is 4 bytes length split in 2 fields:

Expected Packet Number .... This field is written when PCI-DDC receives the last Emitted Packet Number of a message, when it is not a Composed Message.

Received Packet Number .... This field is incremented for each packet received.

Both fields are reset by PCI-DDC when a received message is completed, however the software must reset the entire LRM space once at initialization.

3.3 PCI transactions

PCI-DDC actually uses very few PCI commands, that are listed below:

- PCI-DDC responds to the configuration commands (**Config Read, Config Write**). All registers previously described are read and written by configuration accesses, where PCI-DDC is a fast slave, always ready, that never asks for disconnecting, retrying or aborting a transaction. Multi-data phase are allowed (burst).

- PCI-DDC acts as a master to access the memory: **Memory Write, Memory Read** and **Memory Read Multiple**.

  - Memory Write transaction command is used by PCI-DDC to write a packet, the LRM and the LMI.
  - Memory Read command is used by PCI-DDC to read the LRM and the LPE from the memory.
  - Memory Read Multiple command is used by PCI-DDC to read packets and improve performances.

PCI-DDC is not able to perform back to back transaction.

3.4 Packet Formats

From the point of view of the RCUBE router, the packet format has three parts: header (two bytes), data (N bytes) , end of packet (one byte).

← **Propagation direction**	Header	. . . Data . . .	End

Figure 3.22: router packet structure

From the point of view of the protocol implemented by PCI-DDC, the packet format is more complicated: There are two packet formats, long and short. The long format is used for the standard messages, the short format is used for the short messages and the composed messages. The packet length is always a multiple of 4 bytes. One word is four bytes. A byte contains 9 bits: One bit is used to tag control characters

Sent or received byte order, for every word, is byte 0 first. Packet begins, therefore, with the High Routing Part char and ends with the **EP** character. A bullet means an undefined value.

	Byte 3	Byte 2	Byte 1	Byte 0
Header	*BP*	•	Low Routing Part	High Routing Part
	Message Flags	Message Identifier		
	Page Remote Start Address			1 1
	Header CRC			
Data	byte 1	•	•	•
	byte 5	byte 4	byte 3	byte 2
	byte 9	byte 8	byte 7	byte 6
	•	•	byte 11	byte 10
End	*ED*	0 | 0 0 0 0 0 | 0 1	Expected Packet Number	
	Data CRC			
	EP	•	•	•

Last Byte Address
CRC disable
Last Packet

Figure 3.23: Long packet format

Routing Part (RP) .... is the destination Node Number It is a 16-bit number.

Begin Packet (BP) .... is a control character: `1 0010 1110 (12Eh)`

Message Flags .... same as in the Page Descriptor.

Message Identifier (MI) .... is the message identifier.

Page Remote Start Address (PRSA). is the physical address where data will be placed in the remote memory

Header CRC .... is a CRC code that protects the 3 first words.

Data .... is the data field. Data words are aligned for the receiver. It means that every word in the packet is a word in the remote memory (eg: bytes 2 to 5, bytes 6 to 9). The two PRSA last bit points to the first valid byte in the first word. In the example, 11 points to byte 1. The valid bytes in the last word are defined in the end packet flags.

Expected Packet Number .... This field is undefined, except for the last packet in a standard message. In this case, this number is the transmitted packet number which is the Expected Packet Number for the receiver.

Last Packet .... is set on the last packet for a given message.

CRC disable .... Whenever this field is full one (11111), the data CRC is ignored for the current packet. This is used for packet built by another component that cannot compute CRC on the fly. The header CRC is not disabled.

Last Byte Address .... is the address byte on the last data word. In the example, this address is 1: byte 11 is the last valid byte.

End of Data(ED) .... is a control character: `1 0010 1111 (12Fh)`

data CRC .... is a CRC code that protects the data.

End of Packet(EP) .... is a control character: `1 0010 1011 (12Bh)`

	Byte 3	Byte 2	Byte 1	Byte 0
	BP	•	Low Routing Part	High Routing Part
Header	Message Flags	Message Identifier		
data	Data 1			
	Data 2			
End	CRC			
	EP	•	•	•

Figure 3.24: Short packet format

3.5 PCI-DDC initialization

PCI-DDC initialization is quite complex since the software must not only configure PCI-DDC but can initialize the RCUBE routing tables (See RCUBE specification for details).

- RCUBE needs to have its NRESET pin low to be configured. It is therefore necessary to control this pin from PCI-DDC (R3 init register).
- The second difficulty is start the throughput control between PCI-DDC and RCUBE. It is therefore necessary to begin independently PCI-DDC transmission module (TX init register), PCI-DDC reception module (RX init register) and RCUBE (R3 init register).
- The third difficulty is to forbid PCI-DDC transmission while initialization is not completed, therefore Bus Master bit set ends.

The procedure begins with the RST_N signal assertion or, what is equivalent, by writing one in the Soft Reset register (see PCI-DDC command register). As soon as the RST_N signal is disasserted or one cycle after the Soft Reset register set, PCI-DDC is ready to be configured.

The result of the reset action is:

- flush the FIFO queues,
- set an idle state in the finite state machines,
- set default values in the configuration registers,
- reset (means switch off) RCUBE (R3 init to 0) and the PCI-DDC communication modules (TX init and RX init to 0).

Then, the starting procedure is:

- configure RCUBE routing tables ;
- configure all PCI-DDC registers, except for R3 init, TX init, RX init, and Bus Master ;
- switch on TX module, set TX init ;
- switch on RCUBE, set R3 init ;
- switch on RX module, set RX init ;
- allows PCI-DDC to become master, set Bus Master.

CHAPTER 4
PCI-DDC pins

This chapter details the PCI-DDC pins. The electric type, the direction and a short description are given for each signal. This chapter ends with the package pin assignment.

4.1 Overview

PCI-DDC has 92 logic pins and 28 supply pins in a Quad flat Pack (QFP) 120 pins package.

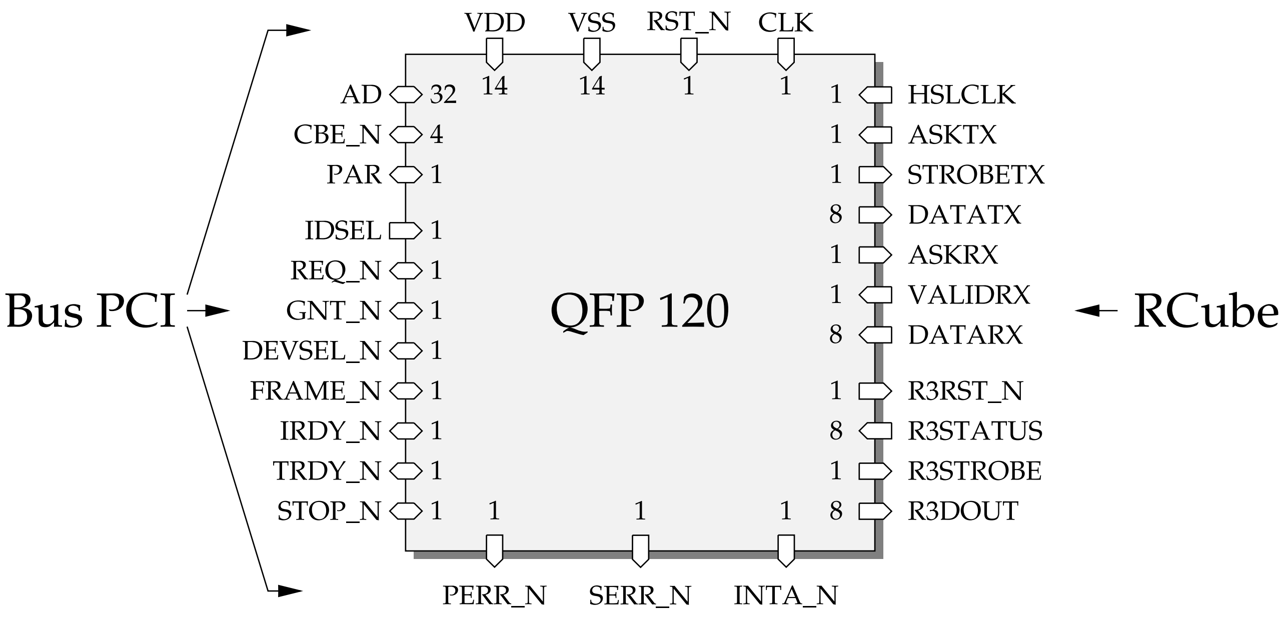


Figure 4.1: PCI-DDC pin overview

The signal electric type are those defined in PCI specification 2.1 page 8, that are:

IN .... Input : standard input signal.

OUT .... Output : standard output signal.

TS .... Tri-State : input-output Tri-State signal.

STS .... Sustained Tri-State : is an active low tri-signal owned and driven by one and only one agent at a time, which must drive an high level before let it float. There is a central weak-pullup to sustain the inactive state.

OD .... Open Drain: is a kind of wired-OR shared by several agents. It is a active low signal. There is a central weak-pullup to sustain the inactive state.

Note that the active low-signal names are suffixed with the _N chars, rather than the # char (PCI specification) because the # char is not allowed by our modeling language (VHDL). Assert a signal means drive the active state, that is 0 for an _N suffixed signal else 1. The following table gives a signal overview. Signals are detailed in sections below.

Table 4.1: PCI-DDC pin list

AD (31 downto 0)	TS	Address and Data (multiplexed on PCI bus)	p. 30
CBE_N (3 downto 0)	TS	Commands and Byte Enables	
PAR	TS	Even PARity on AD and CBE_N	
IDSEL	IN	Initialization Device SELect used to configure PCI-DDC	p. 31
REQ_N	TS	PCI-DDC REQuest to the bus arbiter	
GNT_N	IN	GraNT signal from the bus arbiter	
DEVSEL_N	STS	PCI DEVice (memory) SELect	
FRAME_N	STS	indicate the beginning and duration of a PCI access	
IRDY_N	STS	PCI Initiator ReaDY	
TRDY_N	STS	PCI Target ReaDY	
STOP_N	STS	current PCI target STOP request	
CLK	IN	33 Mhz PCI clock	p. 32
RST_N	IN	PCI reset signal	
PERR_N	STS	PCI data Parity ERRor report	
SERR_N	STS	PCI address and System ERRor report	
INTA_N	OD	PCI interrupt signal	
HSLCLK	IN	80 Mhz RCUBE clock	p. 32
ASKTX	IN	flux control: network asks for data from PCI-DDC	
STROBETX_N	OUT	transmitted data strobe	
DATATX (8 downto 0)	OUT	transmitted data to the network	
ASKRX	OUT	flux control: PCI-DCC asks for data from the network	
VALIDRX	IN	received data validity signal	
DATARX (8 downto 0)	IN	received data from the network	
R3RST_N	OUT	RCUBE reset signal	p. 33
R3STATUS(7 downto 0)	IN	RCUBE link status	
R3STROBE	OUT	RCUBE configuration data strobe	
R3DOUT(7 downto 0)	OUT	RCUBE configuration data	
TEST	IN	Test mode	p. 34
VDD	IN	Power (+5v)	p. 34
VSS	IN	Ground (0v)	

4.2 PCI Address and Data bus

AD (31 downto 0) .... **TS** **+5v**

Address and Data are multiplexed. All PCI transactions begins with an address phase followed by one or several data phases.

The address phase is the first cycle when FRAME_N is asserted. The Address is always a word address (AD(1 downto 0)='00'). AD(31) is the most significant bit, AD(0) is the least significant bit. AD(31 downto 24) is the most significant byte, AD(7 downto 0) is the least significant byte.

CBE_N (3 downto 0) ...... **TS** **+5v**

Commands and Bytes Enable. During the address phase, CBE_N is the command type. PCI-DDC uses or reconize 5 PCI commands, thus 5 CBE_N values:

1010 **Configuration Read** as target from the bridge.

1011 **Configuration Write** as target from the bridge.

0110 **Memory Read** as master to read LRM and LPE structures.

1100 **Memory Read Multiple** as master to read pages.

0111 **Memory Write** as master to write the memory.

PCI-DDC does not map memory or I/O space. It is thus never as target for such commands.

During a data phase, CBE_N contains AD Byte Enables. CBE_N(3) is asserted if AD(31 downto 24) is valid. CBE_N(0) is asserted if AD(7 downto 0) is valid.

PAR....................... **TS** **+5v**

Parity. AD, CBE_N et PAR must have a even number of one.

4.3 PCI control signals

The PCI protocol needs several signals, to ask for the bus, to define the transaction beginning and duration, to control the data throughput and to forgive a transaction.

IDSEL .... **IN** **+5v**

Initialization Device SELect. That signal selects PCI-DDC for read and write configurations.

REQ_N .... **TS** **+5v**

PCI-DDC asserts that signal to ask the arbiter for the bus. It is disasserted when RST_N is asserted.

GNT_N................... **IN** **+5v**

Grant signal from the bus Arbiter.

DEVSEL_N............... **STS** **+5v**

DEVice SELect signal asserts by the agent that has decode its address as target.

FRAME_N................ **STS** **+5v**

Cycle Frame is driven by the current master to indicate the beginning and duration of an access.

IRDY_N .... **STS** **+5v**

Initiator ReaDY indicates the initiating agent's ability to complete the current transaction phase.

TRDY_N .... **STS** **+5v**

Target ReaDY indicates the target agent's ability to complete the current transaction phase.

STOP_N .... **STS** **+5v**

Stop indicates the current target is requesting the master to stop the current transaction.

4.4 PCI special signals

CLK .... **IN** **+5v**

PCI 33 Mhz clock signal. All PCI signals are synchronous apart from INTA_N and RST_N signals.

RST_N .... **IN** **+5v**

Asynchronous initialization signal of PCI-DDC.

PERR_N .... **STS** **+5v**

Parity error signal to only report data parity error.

SERR_N .... **OD** **-**

Parity error signal to only report address parity error.

INTA_N .... **OD** **-**

Interrupt signal uses to report events of PCI-DDC such as errors, transmitted page or received message.

4.5 Network IN/OUT ports

Those port are connected with RCUBE to transmit and received data. They works with the RCUBE clock which is different from the PCI clock. PCI-DDC manages with the asynchronism thanks to internal asynchronous FIFO queues.

HSLCLK .... **IN** **+3.3v**

Clock used for data transmission and data reception between PCI-DDC and RCUBE. PCI-DDC and RCUBE must have the same clock source. HSLCLK pin must be connected with the RCUBE CKIN pin.

ASKTX .... **IN** **+3.3v**

RCUBE asserts that signal during one HSLCLK period to ask PCI-DDC for data to transmit. ASKTX pin is connected with the RCUBE SENDFCC0 pin.

STROBETX_N .... **OUT** **+5v**

It is the strobe signal for the transmitted data to RCUBE. STROBETX_N pin is connected with the RCUBE DATASTROBE0 pin.

DATATX (8 downto 0) . . . . . **OUT** **+5v**

Transmitted data to RCUBE. DATATX port is connected with the RCUBE DATAIN0 port.

ASKRX . . . . **OUT** **+5v**

PCI-DDC asserts that signal during one HSLCLK period to ask RCUBE for data to receive. ASKRX pin is connected with the RCUBE FCCRECEIVED0 pin.

VALIDRX . . . . **IN** **+3.3v**

PCI-DDC records the incoming data from RCUBE when that signal is asserted. VALIDRX pin is connected with the RCUBE SENDDATA0 pin.

DATARX (8 downto 0) . . . . **IN** **+3.3v**

Incoming data from RCUBE. DATARX port is connected with the RCUBE DATAOUT0 port.

4.6 Configuration and status ports

Those signals allows the software to configure RCUBE and to get its link status. They may be used for another component so long it respects PCI-DDC conditions.

R3RST_N . . . . **OD** **-**

Signal used for RCUBE serial link initialization, and internal FIFO queues reset. Routing tables are not concerned. R3RST_N has to be asserted (at low level) at power on, that is why it is a Open Drain out type. It must have an external weak-pullup to sustain the inactive state.

R3RST_N pin is connected with the RCUBE NRESET and ZCRESET pins.

R3STATUS (7 downto 0) . . **IN** **+3.3v**

Those signals allows the software to know the RCUBE link status that is calibration status. A value is sampled every PCI clock cycle to be written in the R3STATUS configuration register. The value is negated before to be written. R3STATUS port is connected with the RCUBE LINKSTATUS ports.

R3STROBE . . . . **OUT** **+5v**

PCI-DDC asserts that signal during one PCI clock cycle, whenever the software writes in the R3DOUT configuration register. RCUBE uses it to strobe R3DOUT port. R3STROBE pin is connected with the RCUBE CLOCKCONFIG pin.

R3DOUT . . . . **OUT** **+5v**

This port allows the software to configure RCUBE routing tables. It reports the R3DOUT configuration register value. R3DOUT port is connected with the RCUBE SCANINCONFIG port.

4.7 PCI-DDC test mode

TEST .... **IN** **+5v**

TEST pin is used for PCI-DDC test. It should be disasserted in normal used.

4.8 Supply

PCI-DDC is fed with 5v. RCUBE is fed with 3.3v. PCI-DDC is compatible with RCUBE since RCUBE may accept both 5v and 3.3v for I/O signal and PCI-DDC accepts 3.3v signals.

VDD .... **IN** **+5v**

Power from the PCI-bus : 7 pins for the internal logic and 7 for the power stage of the I/O cells.

VSS .... **IN** **+0v**

Ground from the the PCI-bus : 7 for the internal logic and 7 for the poser stage of the I/O cells.

4.9 Pinout

We assume a PQFP cavity up package. The pin numbering from 1 to 120 is represented on the figure bellow. The PCI-DDC/PCI pin order respects the PCI specification recommendation (rev

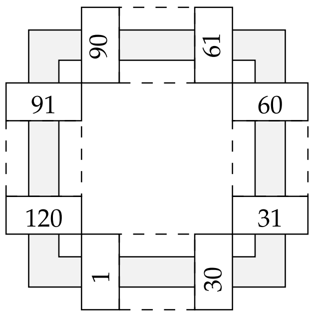


Figure 4.2: PCI-DDC pin numbering

2.1, page 138), except from the clk signal that is placed in order to reduce skew problem inside the component. The PCI-DDC/RCUBE pin are placed in order to avoid track crossing on the PCB, when the WEST side of RCUBE is in front of the NORTH side of PCI-DDC.

SOUTH side		EAST side		NORTH side		WEST side	
1	AD[22]	**31**	AD[10]	**61**	DATARX[0]	**91**	R3STATUS[0]
2	AD[21]	**32**	AD[9]	**62**	DATARX[1]	**92**	R3STATUS[2]
3	**VSSE**	**33**	**VSSI**	**63**	**VDDE**	**93**	**VDDI**
4	AD[20]	**34**	AD[8]	**64**	DATARX[2]	**94**	R3STATUS[4]
5	AD[19]	**35**	CBE_N[0]	**65**	DATARX[3]	**95**	R3STATUS[6]
6	AD[18]	**36**	AD[7]	**66**	DATARX[4]	**96**	R3STATUS[7]
7	**VDDI**	**37**	**VSSE**	**67**	**VSSI**	**97**	**VDDE**
8	AD[17]	**38**	AD[6]	**68**	DATARX[5]	**98**	R3STATUS[5]
9	AD[16]	**39**	AD[5]	**69**	DATARX[6]	**99**	R3STATUS[3]
10	CBE_N[2]	**40**	AD[4]	**70**	DATARX[7]	**100**	R3STATUS[1]
11	**VDDE**	**41**	**VDDI**	**71**	**VSSE**	**101**	**VSSI**
12	FRAME_N	**42**	AD[3]	**72**	DATARX[8]	**102**	INTA_N
13	IRDY_N	**43**	AD[2]	**73**	VALIDRX	**103**	RST_N
14	TRDY_N	**44**	AD[1]	**74**	ASKTX	**104**	GNT_N
15	CLK	**45**	AD[0]	**75**	ASKRX	**105**	REQ_N
16	**VSSI**	**46**	**VDDE**	**76**	**VDDI**	**106**	**VSSE**
17	DEVSEL_N	**47**	TEST	**77**	HSLCLK	**107**	AD[31]
18	STOP_N	**48**	R3STROBE	**78**	DATATX[0]	**108**	AD[30]
19	PERR_N	**49**	R3RST	**79**	DATATX[1]	**109**	AD[29]
20	**VSSE**	**50**	**VSSI**	**80**	**VDDE**	**110**	**VDDI**
21	SERR_N	**51**	R3DOUT[0]	**81**	DATATX[2]	**111**	AD[28]
22	PAR	**52**	R3DOUT[1]	**82**	DATATX[3]	**112**	AD[27]
23	CBE_N[1]	**53**	R3DOUT[2]	**83**	DATATX[4]	**113**	AD[26]
24	**VDDI**	**54**	**VSSE**	**84**	**VSSI**	**114**	**VDDE**
25	AD[15]	**55**	R3DOUT[3]	**85**	DATATX[5]	**115**	AD[25]
26	AD[14]	**56**	R3DOUT[4]	**86**	DATATX[6]	**116**	AD[24]
27	AD[13]	**57**	R3DOUT[5]	**87**	DATATX[7]	**117**	CBE_N[3]
28	**VDDE**	**58**	**VDDI**	**88**	**VSSE**	**118**	**VSSI**
29	AD[12]	**59**	R3DOUT[6]	**89**	DATATX[8]	**119**	IDSEL
30	AD[11]	**60**	R3DOUT[7]	**90**	STROBETX_N	**120**	AD[23]

CHAPTER 5
Features

Manufacturer

PCI-DDC should be use a 0.8μmeter process, two metal, one poly. The first prototype should be made by ATMEL/ES2 with the ecpd07 process for some pieces. The other runs has to be decided yet.

Environment conditions / temperature range

All simulations has been done using the ecpd07 technology files and as best conditions: best process, 0^o (for the junction), 5.25V, and as worst conditions: worst process, 70^o (for the junction), 4.75.

Flexibility and expandability

PCI-DDC is designed to exchange data between a 5-Volt-32-bit-33-MHz-PCI-bus and a 80-MHz RCUBE router. PCI-DDC can fit RCUBE frequencies at least till 80 MHz, but not a 3.3-Volt, or 64-bit or 66-MHz PCI-bus. Those last require another ASIC.

Interchangeability

PCI-DDC is not a COTS but an ASIC, interchangeability is therefore not possible.

Power dissipation

PCI-DDC power dissipation should be less than 2 Watts. A first **estimation** has been done of the power dissipation. It depends of PCI load capacities which should be about 50pF.

$$Power(W) = 0.8 + .01(W/pF) * Load(pF)$$