PCA-8288/8688

Programmer's Guide

Document history		
date	version	changes
17.12.2015	12.2015	the initial release (translated from Czech)

Caution

The TEDIA® products may be used only according to the manufacturer‘s recommendations and precautions given in this document and other general standards and terms and may be used only such a way, that its failure caused by any reason will not be dangerous to any person or property.

Disclaimer

This document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, TEDIA® reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition.

TEDIA® provides this document “as is”, without warranty of any kind, either expressed or implied, including, but not limited to, its particular purpose. TEDIA® reserves the right to make improvements and/or changes to this document, or to the products and/or the programs described in this document, at any time.

Information provided in this document is intended to be accurate and reliable. The reader should contact TEDIA®, if errors are suspected. In no event shall TEDIA® be held liable for any form of damage arising out of or related to this document or the information contained in it.

All brand names and trademarks used in this document are the property of their respective owners.

Manufacturing, sales office, service center, technical support and headquarters:

address: TEDIA spol. s r. o., Zabelska 12, 31211 Plzen, Czech Republic

website: https://www.tedia.eu

phone/e-mail: https://www.tedia.eu/contacts

tech. support: https://www.tedia.eu/support

Table of Contents

1. General information

1.1 Introduction

This Programmer's Guide follows the PCA-8288/8688 card User's Guide (hereinafter all types referred as PCA-8x88) containing ...

- basic technical data,
- description of installation procedure
- and description of the connector pin assignment.

User's Guide is intended for a regular card user who only needs to install card and use it with already created programs.

Unlike the User's Guide, the Programmer's Guide contains ...

- description of the PCI Express controller built into the card,
- description of all functional registers of the card
- and description of register-level programming.

Programmer's Guide therefore enables programming using a system driver with an API offering direct access to the registers (in the case of Windows it is the tedia_ep4gxa.dll library), i.e. creating special programs or custom drivers (e.g. for various SCADA systems or for the Linux operating system).

1.2 Standard height and low-profile card design

DAQ PCI Express TEDIA cards are available in a standard height version (type designation PCA-8x88) and low-profile version (type designation PCA-8x88/LP). With the exception of the different locations of connectors and the applicable accessories, both version are identical and the information contained in this manual is therefore valid for both variants.

1.3 Firmware version

Current firmware version at the time this document was issued:

FPGA - firmware type:	2E	(represented by a value $2E_H$)
FPGA - firmware version:	0.2	(represented by a value 02_H)

The FPGA firmware type is a control number assigned to the standard PCA-8x88 firmware. A different number represents either an incorrect firmware configuration (for example, intended for a different card) or custom firmware.

The FPGA firmware version is an additional parameter that defines the card properties.

Note: *The features described in this document reflect the specified firmware version. Later firmware versions (unless otherwise noted) will be backward compatible with the current version and will include improvements to existing functionality or brand new features.*

1.4 Where to get more information, technical support

Further useful information is available at...

website: https://www.tedia.eu

In doubt, you can contact the manufacturer's technical support:

address:	TEDIA spol. s r. o., Zabelska 12, 31211 Plzen, Czech Republic
phone/e-mail:	https://www.tedia.eu/contacts
tech. support:	https://www.tedia.eu/support

Note: *Although this Programmer's Guide has been carefully reviewed, it can contain errors. If you suspect that some information is listed incorrectly, incompletely or inaccurately, please contact technical support.*

2. PCI Express controller

2.1 Introduction

All PCA-8x88 cards are equipped with a PCI Express bus controller core implemented in an FPGA gate array (i.e. the cards do not use any special PCI Express controller/bridge to the local bus interconnecting I/O peripherals).

The implementation of controller is single-functional (the card thus behaves as one PCI device) with three address spaces (BAR) mapped in the in 32-bit address MEM space.

Note: *Although the card's registers are mapped to the MEM space with 32-bit addressing, the DMA controller (if implemented) supports both 32-bit and 64-bit addressing modes.*

2.2 PCI configuration register space

The table below provides an overview of the selected registers from the PCI configuration register space.

address	register name	PCA-8288 PCA-8288/LP	PCA-8688 PCA-8688/LP
$01_H \div 00_H$	Vendor ID	1760_H (i.e. TEDIA VID)	
$03_H \div 02_H$	Device ID	0860_H	0861_H
08_H	Revision ID	01_H	
$0B_H \div 09_H$	Class Code	118000_H (i.e. PCI class "other data acquisition controller")	
$13_H \div 10_H$	BAR0	functional card's registers (MEM, 16kB, assigned by BIOS)	
$17_H \div 14_H$	BAR1	service purpose registers (firmware update, calibration constants, etc.) (MEM, 16kB, assigned by BIOS)	
$1B_H \div 18_H$	BAR2	registers intended for the kernel part of the operating system driver (MEM, 4kB, assigned by BIOS)	
$1F_H \div 1C_H$	BAR3	unused	
$23_H \div 20_H$	BAR4	unused	
$27_H \div 24_H$	BAR5	unused	
$2D_H \div 2C_H$	Subsystem Vendor ID	1760_H (i.e. TEDIA VID)	
$2F_H \div 2E_H$	Subsystem ID	0001_H	
$3C_H$	Interrupt Line	number of IRQ channel (assigned by BIOS)	
$3D_H$	Interrupt Pin	01_H (INTA)	

What is the information provided by the PCI configuration registers described above ...

- Vendor ID and Device ID are intended for identification of the card type in the system (in case of uncertainty, Subsystem Vendor ID and Subsystem ID may also be used, or Class Code)
- BARx are designed to determine the allocated resources, i.e. the starting address of the register blocks
- The Interrupt Line is designed to determine the current connection of the card's INT signal to the logic IRQ interrupt channel (resp. to the MSI channel in case of this interrupt mode)

2.3 Register's mapping

The following paragraphs describe general information about register mapping.

Register's mapping only in MEM space and not in I/O space as with TEDIA DAQ PCI cards ...

Mapping in I/O space is obsolete and very restrictive (it allows to allocate a total of 255 blocks of 256 bytes size to all PCI devices in the computer) and finds meaningful use only in operating systems (resp. development tools) that do not allow simple 32-bit or 64-bit addressing of MEM space (eg. MS-DOS).

Registers in BAR0 space ...

This BAR space contains all user registers (i.e. registers accessing I/O peripherals of the card).

The following chapters are dedicated, with a few exceptions, exclusively on the description of these registers.

Registers in BAR1/BAR2 space ...

These BAR spaces contain service purpose registers and registers intended for kernel part of the operating system driver.

3. Functional registers

3.1 Registers overview

The tables in the following paragraphs provide an overview of the functional registers implemented in the current firmware version (see Chapter 1). All functional registers described in this chapter are mapped in BAR0 space.

Warning: *All registers, unless explicitly stated otherwise (e.g. registers whose initial value can be defined via the EEPROM contents), have zero values after power-up start or reset.*
However, when program starts, it can not rely on this state because the registers can be set to different values by the previous program; they can be set to defined state either by programming values or by using the CardResetReg register.

Note: *If you are creating a program that supports multiple types of card, it is also recommended to use a document containing tables with comparison of register maps for different types of cards.*

3.2 Splitting the address space into blocks

The table below provides an overview of splitting of the entire BAR0 address space into several blocks according to I/O peripherals; this structure is used by all types of TEDIA DAQ PCI Express card.

offset within BAR0	description of registers block
+0000 ÷ 03FC	registers with 8-bit data (intended for easier migration from DAQ PCI cards)
+0400 ÷ 07FC	registers with 32-bit data (block of DIO ports and edge detection circuits)
+0800 ÷ 13FC	registers with 32-bit data (PCA-8x88 does not use this block)
+1400 ÷ 14FC	registers with 32-bit data (block of analog outputs)
+1500 ÷ 3EFC	registers with 32-bit data (PCA-8x88 does not use this block)
+3F00 ÷ 3FFC	diagnostic registers (common to all types of cards)

3.3 Block of registers with 8-bit data (+0000 ÷ 03FC)

The table below provides a list of registers with 8-bit data.

The registers can be accessed with byte operand to the address specified in the table, or with dword operand, with valid data being transmitted on the lowest eight bits (higher bits are ignored during writing and zeroed during reading).

The program can only access addresses aligned to the dword (i.e. in integer multiple of 4) and it is not recommended to access addresses other than listed in the table.

offset within BAR0	description of register (write)	description of register (read)
+0000	DOUTReg0	DINReg0
+0004	DOUTReg1	DINReg1
+0008	DOUTReg2	DINReg2
+000C	(reserved for DOUTReg3)	DINReg3
+0010	(reserved for DOUTReg4)	DINReg4
+0014	(reserved for DOUTReg5)	DINReg5
+0080	DIOCfgReg	(read back the register value)
+0200	IRQCfgReg	IRQStatusReg
+0204	IRQClrReg	- - -
+0208	TimerReg	TimerReg
+020C	INTEnReg	(read back the register value)
+03F4		CardIDReg
+03F8	- - -	FPGATypeReg
+03FC	- - -	FPGAVerReg

Note: *The register mapping of the first three DIO ports is identical for all TEDIA DAQ PCIe cards.*

3.4 Block of DIO ports and edge detection circuits registers (+0400 ÷ 07FC)

The table below provides a list of registers dedicated for access to DIO ports and signal rising/falling edge detection circuits with the possibility of triggering an interrupt.

The program can only access registers with dword operand at addresses aligned to the dword (i.e. in integer multiple of 4) and it is not recommended to access addresses other than listed in the table.

offset within BAR0	description of register (write)	description of register (read)
+0400	DOUTReg(2-0)	DINReg(2-0)
+0410	DINREReg(2-0)	DINREStatusReg(2-0)
+0414	DINREClrReg(2-0)	- - -
+0418	DINFEReg(2-0)	DINFEStatusReg(2-0)
+041C	DINFEClrReg(2-0)	- - -
+0440	DINREIRQReg(2-0)	(read back the register value)
+0444	DINFEIRQReg(2-0)	(read back the register value)

3.5 Block of analog outputs registers (+1400 ÷ 14FC)

The table below provides a list of registers dedicated for access to registers of analog outputs.

The program can only access registers with dword operand at addresses aligned to the dword (i.e. in integer multiple of 4) and it is not recommended to access addresses other than listed in the table.

offset within BAR0	description of register (write)	description of register (read)
+1400	DAC0Reg	(read back the register value)
+1404	DAC1Reg	(read back the register value)
+1408	DAC2Reg	(read back the register value)
+140C	DAC3Reg	(read back the register value)
+1410	DAC4Reg	(read back the register value)
+1414	DAC5Reg	(read back the register value)
+1418	DAC6Reg	(read back the register value)
+141C	DAC7Reg	(read back the register value)
+14A0	DAC0RegLo	(read back the register value)
+14A4	DAC1RegLo	(read back the register value)
+14A8	DAC2RegLo	(read back the register value)
+14AC	DAC3RegLo	(read back the register value)
+14B0	DAC4RegLo	(read back the register value)
+14B4	DAC5RegLo	(read back the register value)
+14B8	DAC6RegLo	(read back the register value)
+14BC	DAC7RegLo	(read back the register value)
+14C0	DAC0RegHi	(read back the register value)
+14C4	DAC1RegHi	(read back the register value)
+14C8	DAC2RegHi	(read back the register value)
+14CC	DAC3RegHi	(read back the register value)
+14D0	DAC4RegHi	(read back the register value)
+14D4	DAC5RegHi	(read back the register value)
+14D8	DAC6RegHi	(read back the register value)
+14DC	DAC7RegHi	(read back the register value)
+14E0	- - -	DAC0PHYReg
+14E4	- - -	DAC1PHYReg
+14E8	- - -	DAC2PHYReg
+14EC	- - -	DAC3PHYReg
+14F0	- - -	DAC4PHYReg
+14F4	- - -	DAC5PHYReg
+14F8	- - -	DAC6PHYReg
+14FC	- - -	DAC7PHYReg

3.6 Block of diagnostic registers (+3F00 ÷ 3FFC)

The table below provides a list of registers dedicated for access to diagnostic and identification functions.

The program can only access registers with dword operand at addresses aligned to the dword (i.e. in integer multiple of 4) and it is not recommended to access addresses other than listed in the table.

offset within BAR0	description of register (write)	description of register (read)
+3FE0	CardResetReg	CardResetStatusReg
+3FF0	- - -	CardIDReg
+3FF4	- - -	CardSerNrReg
+3FF8	- - -	FPGATypeReg
+3FFC	- - -	FPGAVerReg

4. Registers intended for digital inputs/outputs ports

4.1 Introduction

The following sections describe the registers related to digital inputs and outputs (see the overview in Chapter 3).

The DIO registers can be divided into a group of data registers...

DINReg0, ...2	three 8-bit digital port input registers (sets of signals DIO00÷07, DIO08÷15 and DIO16÷23)
DINReg(2-0)	32-bit digital port input register (combines DINReg0, DINReg1 and DINReg2)
DOUTReg0, ...2	three 8-bit digital port output registers (sets of signals DIO00÷07, DIO08÷15 and DIO16÷23)
DOUTReg(2-0)	32-bit digital port output register (combines DOUTReg0, DOUTReg1 and DOUTReg2)

and configuration register...

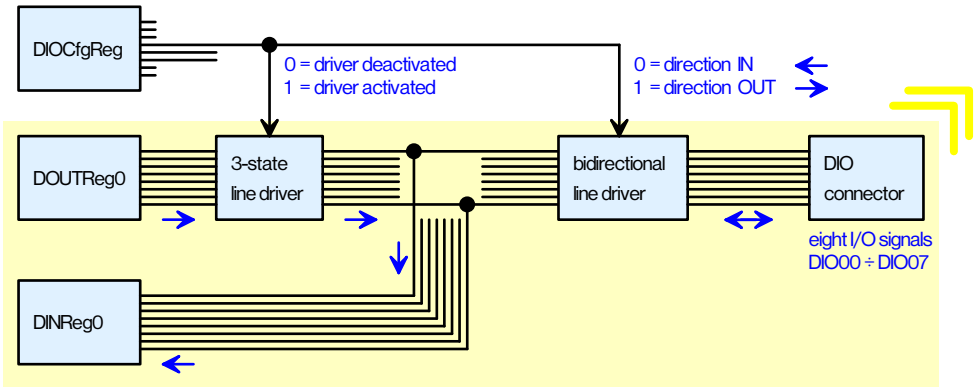
DIOCfgReg	register for configuration of direction DIO ports (i.e. selection of input or output port)

Note: *Initial value (e.g. after power-up or soft-reset using the CardResetReg register) of all registers mentioned above can be defined via the EEPROM contents. Because the initial values stored in the EEPROM can be modified by the configuration program, the user can define the DIO ports status programmed immediately after turning on the computer without delay until the program starts.*

4.2 Digital port functionality

All three ports are designed as bidirectional, i.e. each port (= eight digital signals) can be individually set as input or output. The current state of the port can be obtained by reading the DINx register; in the case of configuration as an input port, the status of input signals is read, in the case of configuration as an output port, current data written to the output register are read back.

Further details can be seen in the picture below (only 8-bit port DIO0 is drawn, the part with the yellow background is included in the card three times, i.e. separately for DIO0, DIO1 and DIO2 port).



Location of ports on the card connectors

All three ports (i.e. DIO0, DIO1 and DIO2) are connected to the KX1 - KX3 connectors located on the back edge of the card (they are accessible via adapter cable), the next three ports (i.e. DIO0, DIO1 and DIO2) are connected to the D-Sub 25 connector located on the card bracket.

Note: *The register mapping of the three DIO ports described above is identical for all TEDIA DAQ PCIe cards.*

4.3 Registers DINReg0, ... , DINReg2 [RD]

These registers are used to read the status of the digital port, each bit of the register accesses one signal of the 8-bit digital port (bits D0 of these registers access the signals DIO00/08/16; bits D7 access DIO07/15/23).

In the case the port is configured as an input port, the register provides a current status of input signals; in the case the port is configured as an output port, current data written to the output register are read back.

4.4 Register DINReg(2-0) [RD]

This register is an an alternative to the 8-bit registers described in the previous paragraph and they are used to read the status of three digital ports at once (functionality remains identical to 8-bit registers).

The DINReg(2-0) register combines DINReg0, DINReg1 and DINReg2 (bits D00÷D23 provide a current value of ports DIO00÷DIO23, the highest eight bits are permanently zeroed).

4.5 Registers DOUTReg0, ... , DOUTReg2 [WR]

These registers are used to control the status of the output digital port, each bit of the register defines one signal of the 8-bit digital port (bits D0 of the registers control the signals DIO00/08/16; bits D7 control DIO07/15/23).

In the case the port is configured as an input port, data can be written to the DOUTReg register, but its contents do not affect the state of the signals. In the case the port is configured as an output port, the current contents of this register defines the status of the output signals.

Writing to these registers is allowed even if the port is set as input, their contents will be transferred to the signals immediately after the port is switched to output mode.

4.6 Register DOUTReg(2-0) [WR]

This register is an alternative to the 8-bit registers described in the previous paragraph and it is used to control the contents of three digital ports at once (functionality remains identical to 8-bit registers).

The DOUTReg(2-0) register combines DOUTReg0, DOUTReg1 and DOUTReg2 (bits D00÷D23 define the status of the output signal DIO00÷DIO23, the highest eight bits are ignored).

4.7 Register DIOCfgReg [WR/RD]

This register is used to configure bidirectional DIO ports as input or output.

D7	D6	D5	D4	D3	D2	D1	D0
RSRV					DIR2	DIR1	DIR0

DIR0 direction control of the DIO0 port
- 0 the outputs of the DOUT register are deactivated (i.e. the port works as an input)
- 1 the outputs of the DOUT register are activated (i.e. the port works as an output)

DIR1 direction control of the DIO1 port
- 0 the outputs of the DOUT register are deactivated (i.e. the port works as an input)
- 1 the outputs of the DOUT register are activated (i.e. the port works as an output)

DIR2 direction control of the DIO2 port
- 0 the outputs of the DOUT register are deactivated (i.e. the port works as an input)
- 1 the outputs of the DOUT register are activated (i.e. the port works as an output)

RSRV reserved (to ensure forward compatibility it is recommended to write 0 and ignore readed value)

5. Registers intended for DIO signal edge detection circuits

5.1 Introduction

The following sections describe the registers related to signal edge detection circuits (see the overview in Chapter 3).

List of registers:

DINREReg(2-0)	enable of rising edge detection on DIO00÷DIO23 port signals
DINREStatusReg(2-0)	rising edge event flags on DIO00÷DIO23 port signals
DINREClrReg(2-0)	clear rising edge event flags on DIO00÷DIO23 port signals
DINFEReg(2-0)	enable of falling edge detection on DIO00÷DIO23 port signals
DINFEStatusReg(2-0)	falling edge event flags on DIO00÷DIO23 port signals
DINFEClrReg(2-0)	clear falling edge event flags on DIO00÷DIO23 port signals
DINREIRQReg(2-0)	enable interrupts from edge detection circuits (rising edge detection on DIO00÷DIO23)
DINFEIRQReg(2-0)	enable interrupts from edge detection circuits (falling edge detection on DIO00÷DIO23)

5.2 Registers DINREReg(x-x) and DINFEReg(x-x) [WR]

These registers are intended to enable rising edge detection (DINREReg), resp. falling edge detection (DINFEReg) on DIO port signals.

These registers have significant 24 lowest bits (bit D0 with value 1 enable edge detection on signal DIO00; bit D23 with value 1 enable edge detection on signal DIO23), the highest eight bits are ignored (value 0 recommended to ensure forward compatibility).

5.3 Registers DINREStatusReg(x-x) and DINFEStatusReg(x-x) [RD]

These registers are intended to determine the status of the flags of the edge detection circuits enabled by the DINREReg(x-x) and DINFEReg(x-x) registers.

These registers have significant 24 lowest bits (value 1 on bit D0 indicate active flags related to signal DIO00; value 1 on bit D23 indicate active flags related to signals DIO23), the highest eight bits are permanently zeroed.

5.4 Registers DINREClrReg(x-x) and DINFEClrReg(x-x) [WR]

These registers are intended to clearing selected flags of the edge detection circuits.

These registers have significant 24 lowest bits (bit D0 with value 1 clear flags related to signal DIO00, bit D23 with value 1 clear flags related to signal DIO23), the highest eight bits are ignored (value 0 recommended to ensure forward compatibility).

Writing a value of 1 to a defined bit generates a short pulse, so following write of 0 is not required. Clearing any combination of flags in a single write cycle is allowed.

5.5 Registers DINREIRQReg(x-x) and DINFEIRQReg(x-x) [WR]

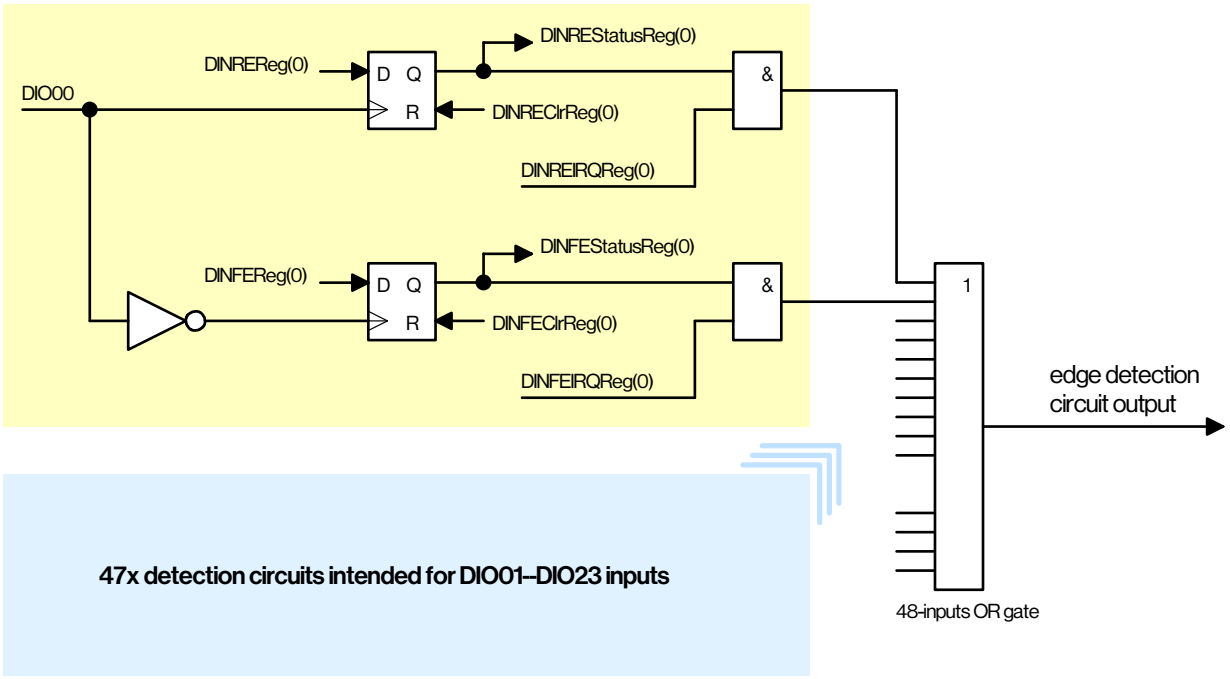
These registers are intended to enable system interrupt event generation related to edge detection circuit flags.

These registers have significant 24 lowest bits (bit D0 with value 1 enable interrupt event generation related to signal DIO00; bit D23 with value 1 enable interrupt event generation related to signal DIO23), the highest eight bits are ignored (value 0 recommended to ensure forward compatibility).

More detailed information can be found in the picture and description given in the following paragraph, resp. in a separate chapter with a description of interrupt circuits.

5.6 Simplified schematic diagram of edge detection circuits register structure

The figure below shows a simplified schematic diagram of the edge detection circuits, the connection with interrupt circuits is described in a separate chapter.



Each DIOxx signal is equipped with identical circuits that allow the independent detection of rising edge, falling edge or both edges on this signal.

The part of the schematics diagram highlighted in yellow shows the circuits dedicated for processing the signal DIO00, the blue marked part shows the identical circuits for the signals DIO01 to DIO23.

All 48 flags (available for reading via the DINREStatus and DINFEStatus registers) allow to trigger an interrupt. Flags are processed by AND gates in the first step (DINREIRQReg and DINFEIRQReg with value 1 allow flags signal to pass through AND gate) and in the second step by OR gate.

As can be seen from the schematics diagram of the interrupt circuits (see the description in a separate chapter), the interrupt of the system is triggered by the first detected edge, i.e. by transitioning the OR gate output from 0 to 1. To further trigger an interrupt, it is therefore necessary for the interrupt program handler to process all interrupt events and then clear all the flags of the edge detection circuits.

6. Registers intended for interrupt handling circuits

6.1 Introduction

The following sections describe the registers related to interrupt handling circuits (see the overview in Chapter 3).

List of registers:

INTEnReg	interconnection of interrupt detection circuits (all registers described in this chapter) with card interrupt generation circuits (both INTA or MSI mode)
IRQCfgReg	enabling of general interrupt sources
IRQStatusReg	statuses of general interrupt sources
IRQClrReg	clearing of general interrupt flags
TimerReg	timestamp generator for periodically triggering interrupts

6.2 Interrupt handling circuit functionality

Interrupt handling circuits have the capability to cause a system interrupt event by one of the sources, or by a selected combination of interrupt sources. The card uses the following interrupt sources:

Timestamp generator

Allows to trigger an interrupt with a selected time period in the range of 1÷254 ms.

Digital inputs - simple mode compatible with DAQ PCI cards

Allows to trigger an interrupt with falling edge of selected DIO port signals.

Digital inputs - edge detection circuits used by DAQ PCI Express cards

Allows to trigger an interrupt by any combination of rising and falling edges on all DIO port signals.

Note: *For proper function of the interrupt circuits (see the schematics diagram on the following pages) it is necessary to consider, that interrupt of the system is triggered by the first event. To further trigger an interrupt, it is therefore necessary for the interrupt program handler to process all interrupt events and then clear all the flags (optional flags of the edge detection circuits).*

6.3 Register INTEnReg [WR]

These registers are dedicated to interconnection of interrupt detection circuits (all registers described in this chapter) with card interrupt generation circuits (both INTA or MSI mode).

D7	D6	D5	D4	D3	D2	D1	D0
INTEN	RSRV						

INTEN activation of INTA/MSI control circuits
- 0 capture register generating control signal INTA or generating MSI is permanently zeroed
- 1 the capture register function is activated, i.e. the card may triggered a system interrupt

RSRV reserved (to ensure forward compatibility it is recommended to write 0)

6.4 Register IRQCfgReg [WR]

These registers are dedicated to enable of general interrupt sources.

D7	D6	D5	D4	D3	D2	D1	D0
RSRV	DIN-X	RSRV	TIM	RSRV	IRQ2	IRQ1	IRQ0

IRQ0 enables the trigger an interrupt derived from the falling edge of digital port DIO00
- 0 the capture register connected with digital input is disabled
- 1 the capture register functionality is enabled

IRQ1 enables the trigger an interrupt derived from the falling edge of digital port DIO08
- 0 the capture register connected with digital input is disabled
- 1 the capture register functionality is enabled

IRQ2 enables the trigger an interrupt derived from the falling edge of digital port DIO16
- 0 the capture register connected with digital input is disabled
- 1 the capture register functionality is enabled

TIM enables the trigger an interrupt derived from timestamp generator
- 0 the capture register connected with timestamp generator is disabled
- 1 the capture register functionality is enabled

DIN-X enables the trigger an interrupt derived from signal edge detection circuits
- 0 the capture register connected with signal edge detection circuits
- 1 the capture register functionality is enabled

RSRV reserved (to ensure forward compatibility it is recommended to write 0)

6.5 Register IRQStatusReg [RD]

This register is dedicated to determine the status of the capture registers enabled by the IRQCfgReg register.

D7	D6	D5	D4	D3	D2	D1	D0
RSRV	DIN-X	RSRV	TIM	RSRV	IRQ2	IRQ1	IRQ0

IRQ0 the current status of the capture register connected with digital input DIO00
- 0 the register is not set, i.e. no falling edge has been detected since the last reset
- 1 the register is set, i.e. an interrupt request event has occurred since the last reset

IRQ1 the current status of the capture register connected with digital input DIO08
- 0 the register is not set, i.e. no falling edge has been detected since the last reset
- 1 the register is set, i.e. an interrupt request event has occurred since the last reset

IRQ2 the current status of the capture register connected with digital input DIO16
- 0 the register is not set, i.e. no falling edge has been detected since the last reset
- 1 the register is set, i.e. an interrupt request event has occurred since the last reset

TIM the current status of the capture register connected with timestamp generator
- 0 the register is not set, i.e. no tome stamp has been detected since the last reset
- 1 the register is set, i.e. an interrupt request event has occurred since the last reset

DIN-X the current status of the capture register connected with edge detection circuits
- 0 the register is not set, i.e. no edge detection circuit event has been detected since the last reset
- 1 the register is set, i.e. an interrupt request event has occurred since the last reset

RSRV reserved (to ensure forward compatibility it is recommended to ignore readed value)

6.6 Register IRQClrReg [WR]

This register is dedicated to clearing selected flags enabled by the IRQCfgReg register.

D7	D6	D5	D4	D3	D2	D1	D0
RSRV	DIN-X	RSRV	TIM	RSRV	IRQ2	IRQ1	IRQ0

IRQ0 resets the capture register connected with digital input DIO00
- 0 idle write, the state of the capture register is not modified
- 1 the capture register is reset (subsequent writing 0 is not required)

IRQ1 resets the capture register connected with digital input DIO08
- 0 idle write, the state of the capture register is not modified
- 1 the capture register is reset (subsequent writing 0 is not required)

IRQ2 resets the capture register connected with digital input DIO16
- 0 idle write, the state of the capture register is not modified
- 1 the capture register is reset (subsequent writing 0 is not required)

TIM resets the capture register connected with timestamp generator
- 0 idle write, the state of the capture register is not modified
- 1 the capture register is reset (subsequent writing 0 is not required)

DIN-X resets the capture register connected with edge detection circuits
- 0 idle write, the state of the capture register is not modified
- 1 the capture register is reset (subsequent writing 0 is not required)

RSRV reserved (to ensure forward compatibility it is recommended to write 0)

6.7 Register TimerReg [WR/RD]

This register is dedicated to control a timestamp generator intended mainly to periodically trigger a system interrupt.

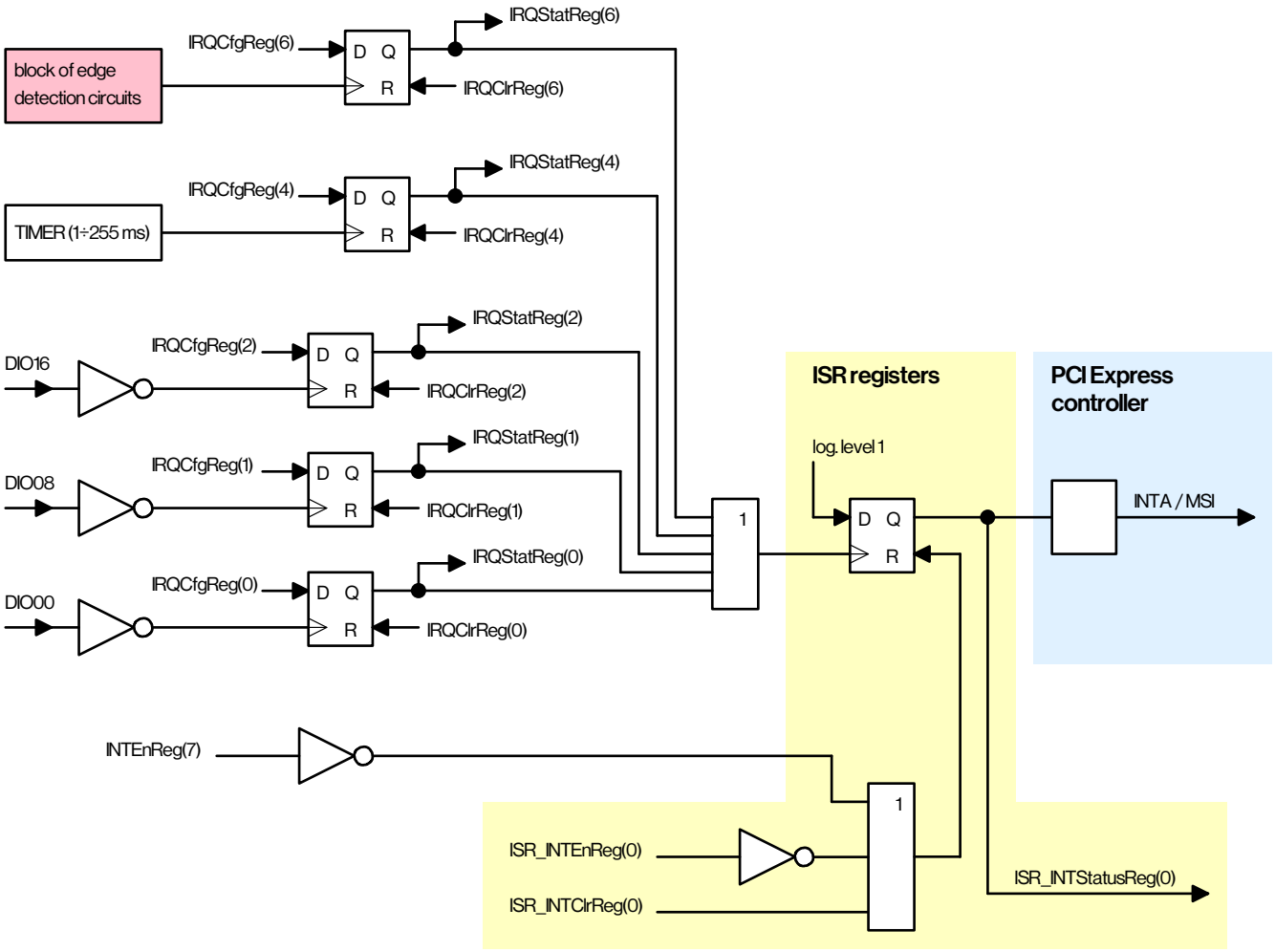
The initial value of the register is zero and the timestamp generator is stopped. By writing a non-zero value, the generator is started, the period is defined by the written value in milliseconds (up to 254 ms). By writing a zero value, the generator is stopped again.

The register is also important for reading (this register provides a current value of the timestamp counter incremented from zero every millisecond to the entered value reduced by one).

For example, by writing a value of 100, the first interrupt is triggered 100 ms after writing to the register and then every 100 ms. The values 0, 1, ... , 98, 99, 0, 1, ..., are provided by reading this register, the system interrupt is triggered at the moment of transition from 99 to 0.

6.8 Simplified schematics diagram of interrupt handling circuits register structure

The figure below shows a simplified schematic diagram of the interrupt handling circuits and interconnection with control/status registers; the edge detection circuits (highlighted in red) are described in a separate chapter.



Note: *All three ISR_*** registers are mapped in BAR2 space and their description is out of focus of this manual.*

From a general point of view - the circuits highlighted with yellow background must be controlled within the ISR (i.e. in the OS kernel), the others can be controlled within the user program, within application driver (e.g. in the case of Windows, the DLL with an API containing abstract high-level functions) or also within the ISR.

The ISR_INTEnReg(0), ISR_INTClrReg(0) and ISR_INTStatusReg(0) signals are implemented identically on all TEDIA DAQ PCIe cards and allow to unify the ISR. The ISR_INTEnReg(0) signal is set to 1 after a power-up start or reset, therefore control via INTEnReg(7) can be also used to create a user specific kernel driver without the need for knowledge of mapping ISR registers in BAR2 space.

Internal solution of the standard tedia_ep4gxa driver for Windows fully complies with the information described above; the ISR_INTEnReg(0) and ISR_INTClrReg(0) signals are controlled from the kernel part of this driver, as the only status information is used ISR_INTStatusReg(0). The ISR is therefore completely independent of the type of card.

Note: *Detailed information on interrupt handling register control is provided in the document "DAQ PCI/PCIe card's Windows system driver Programmer's Guide".*

7. Registers intended for analog outputs

7.1 Introduction

The following sections describe the registers related to analog outputs circuits (see the overview in Chapter 3).

List of registers:

DACxReg	eight registers defining the value of the "xth" analog output (8000_H as default)
DACxRegLo	eight registers defining the lowest valid value written to DACxReg (0_H as default)
DACxRegHi	eight registers defining the highest valid value written to DACxReg ($FFFF_H$ as default)
DACxPHYReg	eight auxiliary registers providing the real data written to the D/A converters

Note: *Initial value (e.g. after power-up or soft-reset using the CardResetReg register) of all registers mentioned above can be defined via the EEPROM contents. Because the initial values stored in the EEPROM can be modified by the configuration program, the user can define the DIO ports status programmed immediately after turning on the computer without delay until the program starts.*
As can be seen from the description above, the factory default values of the registers allow you to set the analog output signal in the full range of ±10 V and the initial signal value is je 0 V.

7.2 Analog outputs functionality

The PCA-8x88 cards contain eight 12-bit D/A converters (PCA-8288) or eight 16-bit D/A converters (PCA-8688), the functionality and register mapping are identical for both type of cards.

All eight D/A converters operate with a single range of approximately ±10.2 V calibrated by linear conversion of ax+b from DACxReg register data to the real range of ±10 V of analog outputs (DACxPHYReg). Calibration constants are stored in the card's EEPROM at the factory.

Analog outputs therefore allow generating a signal in the range of ±10 V by writing to the corresponding register. For applications requiring a different range, the card is additionally equipped with registers allowing limiting the output signal to a smaller range; the data written to the register is compared with the minimum/maximum allowed value (DACxRegLo and DACxRegHI registers)and if it is outside the allowed range, the written value is replaced by the limit value.

7.3 Registers DACxReg [WR]

The contents of these eight registers define the output voltage of the eight analog outputs.

The written data are valid in the range 0÷65535 (i.e. 16-bit data), where the values...

0 (i.e. 0_H)	represents the output voltage -10 V
32768 (i.e. 8000_H)	represents the output voltage 0 V
65535 (i.e. $FFFF_H$)	represents the output voltage +10 V (exactly 32767/32768 * 10 V)

The range of valid data can be limited using the DACxRegLo and DACxRegHi registers.

7.4 Registers DACxRegLo and DACxRegHi [WR]

The contents of these registers define the minimum and maximum allowed values ??written to the DACxReg registers.

When attempting to write a value smaller than DACxRegLo to the DACxReg register, the DACxRegLo value is written to the register, and when attempting to write a value greater than DACxRegHi, the DACxRegHi value is written.

7.5 Registers DACxPHYReg [RD]

The content of these auxiliary registers provides the value actually written to the D/A converters (i.e. the value of DACxReg after calibration conversion ax+b) and is useful for verifying the operation of calibration conversions.

8. Diagnostic registers (common to all card types)

8.1 Introduction

The following sections describe diagnostic registers (see the overview in Chapter 3).

List of registers:

CardResetReg	set all registers of the card to a defined state (identical to state after power-up)
CardResetStatusReg	status of running the procedure of setting all registers to the defined state
CardSerNrReg	provides a unique card serial number
CardIDReg	provides the status of the dual DIP switch (allows to differentiate up to 4 cards)
FPGATypeReg	provides the FPGA firmware type value
FPGAVerReg	provides the FPGA firmware version value

8.2 Register CardResetReg [WR]

Writing the value $5043384B_H$ to this register causes an immediate reset (i.e. set to zero, unless otherwise stated in the register description) all registers except all DOUTRegX registers and DIOCfgReg register.

Immediately after resetting, the preconfigured content is read from the on-board EEPROM and stored to the DOUTRegX and DIOCfgReg registers; this procedure typically takes 1 ms and its progress is signaled by the CardResetStatusReg register. All EEPROM values mentioned above can be modified by the free available configuration program.

Warning: *During the ongoing register setup the program may only access the CardResetStatusReg register.*

8.3 Register CardResetStatusReg [RD]

This register provides an access to flag indicating status of the procedure of setting all registers to the defined state triggered by writing to CardResetReg register.

The register has only the lowest bit significant (all others are permanently zero); a status bit with a value of 1 signals the ongoing setting of the card registers, a value of 0 then corresponds to the idle state.

8.4 Register CardSerNrReg [RD]

This register provides an access to the unique serial number of the card (32-bit integer format).

8.5 Register CardIDReg [RD]

This register provides an access to the status of the dual DIP switch (allows to differentiate up to 4 cards each other).

The register is mapped at two addresses, status is passed on the lowest two bits, the upper six bits (or 30 bits) are permanently zeroed.

8.6 Register FPGATypeReg [RD]

This register provides an access to the FPGA firmware type value in the range 0 to 255.

The register is mapped at two addresses, status is passed on the lowest eight bits, the upper 24 bits are permanently zeroed.

Note: *The value of the standard card firmware type was mentioned in Chapter 1.*

8.7 Register FPGAVerReg [RD]

This register provides an access to the FPGA firmware version value in the range 0 to 255.

The register is mapped at two addresses, status is passed on the lowest eight bits, the upper 24 bits are permanently zeroed.

Note: *The value of the standard card firmware version was mentioned in Chapter 1.*

9. Registers in address spaces BAR1 a BAR2

9.1 Introduction

While the previous chapters described, with a few exceptions, the functional registers in the address space BAR0, the following paragraphs will be dedicated to the registers in the address spaces BAR1 and BAR2.

Warning: *The registers mapped in the BAR1 and BAR2 address spaces are subject to change depending on the firmware version and, unlike the functional registers in BAR0, backward or forward compatibility is not guaranteed. Therefore, the software that uses these registers must modify its functions not only by the card type, but also by the content of the FPGATypeReg and FPGAVerReg registers.*

9.2 Address space BAR1

The address space BAR1 provides an access to service registers (interface for FPGA firmware update, calibration constants, configurable start-up settings of registers, ...) and their description is out of focus of this manual.

Note: *The full specification of BAR1 registers is subject to NDA and can be provided only in justified cases.*

9.3 Address space BAR2

The address space BAR1 provides an access to registers intended for kernel part of the operating system driver (ISR routine regisers, DMA registers, ...) and their description is out of focus of this manual.

Note: *The full specification of BAR1 registers is subject to NDA and can be provided only in justified cases.*