PCT-8303/8306
PCT-8360/8363

Programmer's Guide

Document history		
date	version	changes
21.12.2015	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 PCT-8303/8306/8360/8363 card User's Guide (hereinafter all types referred as PCT-83xx) 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 PCT-83xx) and low-profile version (type designation PCT-83xx/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:	2D	(represented by a value $2D_H$)
FPGA - firmware version:	0.2	(represented by a value 02_H)

The FPGA firmware type is a control number assigned to the standard PCT-83xx 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 PCT-83xx 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	PCT-8303 PCT-8303/LP	PCT-8306 PCT-8306/LP	PCT-8363 PCT-8363/LP	PCT-8360 PCT-8360/LP
$01_H \div 00_H$	Vendor ID	1760_H (i.e. VID TEDIA)			
$03_H \div 02_H$	Device ID	0810_H	0811_H	0812_H	0820_H
08_H	Revision ID	118000_H (i.e. PCI class "other data acquisition controller")			
$0B_H \div 09_H$	Class Code	functional card's registers (MEM, 16kB, assigned by BIOS)			
$13_H \div 10_H$	BAR0	service purpose registers (firmware update, calibration constants, etc.) (MEM, 16kB, assigned by BIOS)			
$17_H \div 14_H$	BAR1	registers intended for the kernel part of the operating system driver (MEM, 4kB, assigned by BIOS)			
$1B_H \div 18_H$	BAR2	unused			
$1F_H \div 1C_H$	BAR3	unused			
$23_H \div 20_H$	BAR4	unused			
$27_H \div 24_H$	BAR5	1760_H (i.e. TEDIA VID)			
$2D_H \div 2C_H$	Subsystem Vendor ID	0001_H			
$2F_H \div 2E_H$	Subsystem ID	number of IRQ channel (assigned by BIOS)			
$3C_H$	Interrupt Line	01_H (INTA)			
$3D_H$	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 ÷ 0FFC	registers with 32-bit data (PCT-83xx does not use this block)
+1000 ÷ 10FC	registers with 32-bit data (block of IRC counters including min/max detectors)
+1100 ÷ 11FC	registers with 32-bit data (block of SSI interfaces)
+1000 ÷ 3EFC	registers with 32-bit data (PCT-83xx 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
+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 IRC counters and min/max detectors registers (+1000 ÷ 10FC)

The table below provides a list of registers dedicated for access to registers of IRC counters and min/max detectors.

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.

The register block on addresses +1000 ÷ 101C contains the registers of the first IRC counter and related config/status functions (i.e. IRCCNT0SetReg, IRCCNT0CWReg, IRCCNT0RngReg, IRCCNT0CWReg, IRCCNT0StatReg, IRCCNT0MinReg and IRCCNT0MaxReg). The analogous structure of the registers for IRC counters IRCCNT1 - IRCCNT5 is mapped to following addresses and these register blocks have exactly the same functionality as the registers block of IRCCNT0.

offset within BAR0	description of register (write)	description of register (read)
+1000	IRCCNT0SetReg	IRCCNT0StrReg
+1004	IRCCNT0RngReg	- - -
+1008	- - -	- - -
+100C	- - -	- - -
+1010	IRCCNT0CWReg	IRCCNT0StatReg
+1014	- - -	- - -
+1018	- - -	IRCCNT0MinReg
+101C	- - -	IRCCNT0MaxReg
+1020 ÷ +103C	registers of IRCCNT1 (structure analogous to block +1000 ÷ +101C)	
+1040 ÷ +105C	registers of IRCCNT2 (structure analogous to block +1000 ÷ +101C)	
+1060 ÷ +107C	registers of IRCCNT3 (structure analogous to block +1000 ÷ +101C)	
+1080 ÷ +109C	registers of IRCCNT4 (structure analogous to block +1000 ÷ +101C)	
+10A0 ÷ +10BC	registers of IRCCNT5 (structure analogous to block +1000 ÷ +101C)	
+10C0	IRCCNTEnReg	(read back the register value)
+10C4	IRCCNTCtrlReg	- - -
+10C8	IRCCNTMinMaxEnReg	(read back the register value)
+10CC	IRCCNTMinMaxCtrlReg	- - -

The table above shows the registers for a set of six IRC counters, however, their implementation depends on the number of IRC counters of the card type.

interface	PCT-8303 (.../LP)	PCT-8306 (.../LP)	PCT-8363 (.../LP)	PCT-8360 (.../LP)
number of IRC counters	3 (CNT0÷CNT2)	6 (CNT0÷CNT5)	3 (CNT0÷CNT2)	0
number of SSI interfaces	0	0	6 (SSI0÷SSI5)	6 (SSI0÷SSI5)

3.6 Block of SSI interfaces registers (+1100 ÷ 11FC)

The table below provides a list of registers dedicated for access to registers of SSI interfaces.

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.

The register block on addresses +1100 ÷ 111C contains the registers of the first SSI interface and related config/status functions (i.e. SSI0CfgReg and SSI0StrReg). The analogous structure of the registers for SSI interfaces SSI1 - SSI5 is mapped to following addresses and these register blocks have exactly the same functionality as the registers block of SSI0.

offset within BAR0	description of register (write)	description of register (read)
+1100	- - -	SSI0StrReg
+1104	- - -	- - -
+1108	- - -	- - -
+110C	- - -	- - -
+1110	SSI0CfgReg	(zpětné čtení)
+1114	- - -	- - -
+1118	- - -	- - -
+111C	- - -	- - -
+1120 ÷ +113C	registers of SSI1 (structure analogous to block +11100 ÷ +111C)	
+1140 ÷ +115C	registers of SSI2 (structure analogous to block +11100 ÷ +111C)	
+1160 ÷ +117C	registers of SSI3 (structure analogous to block +11100 ÷ +111C)	
+1180 ÷ +119C	registers of SSI4 (structure analogous to block +11100 ÷ +111C)	
+11A0 ÷ +11BC	registers of SSI5 (structure analogous to block +11100 ÷ +111C)	
+11C0	SSICfgReg	(read back the register value)
+11C4	SSICtrlReg	- - -

The table above shows the registers for a set of six SSI interfaces, however, their implementation depends on the number of SSI interfaces of the card type.

interface	PCT-8303 (.../LP)	PCT-8306 (.../LP)	PCT-8363 (.../LP)	PCT-8360 (.../LP)
number of IRC counters	3 (CNT0÷CNT2)	6 (CNT0÷CNT5)	3 (CNT0÷CNT2)	0
number of SSI interfaces	0	0	6 (SSI0÷SSI5)	6 (SSI0÷SSI5)

3.7 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...

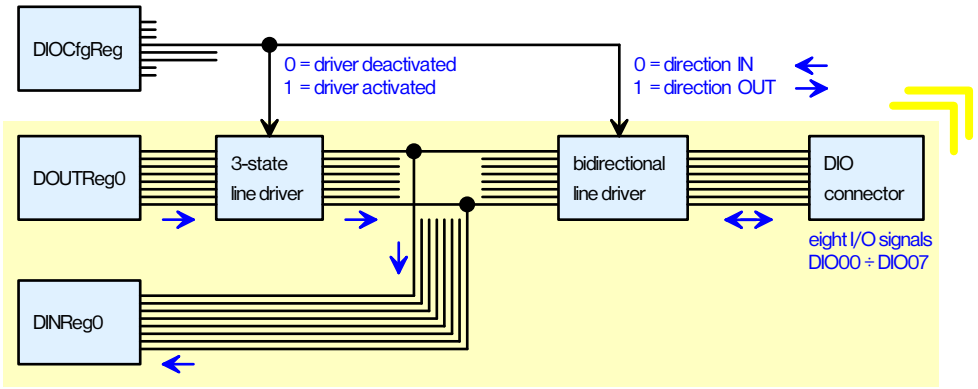
DIOCfgReg	register for configuration of direction DIO ports (i.e. selection of input or output port)

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]

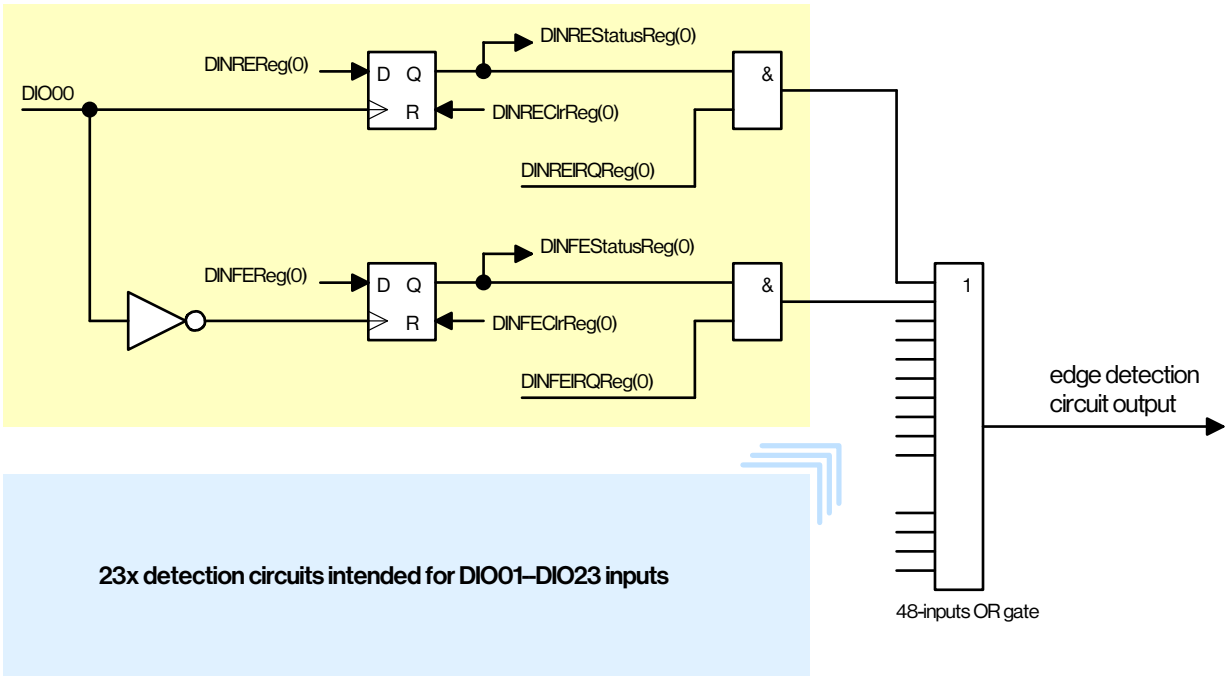
These registers are intended to enable system interrupt event generation related to edge detection circuit flags.

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 intended 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 intended 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 intended 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 intended 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 intended to control a timestamp generator intended mainly to periodically trigger a system interrupt.

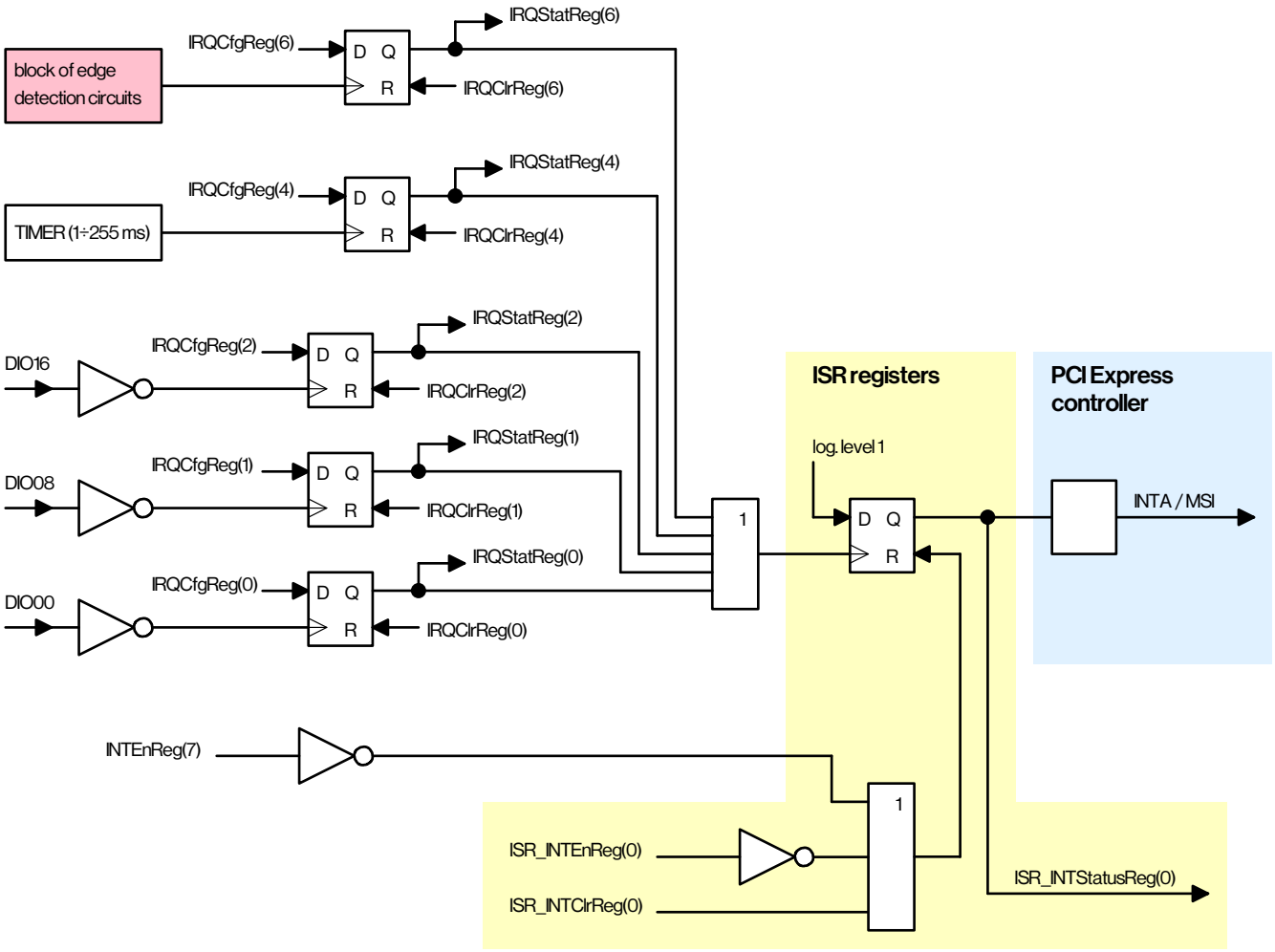
The initial value of the register is zero and the timestamp generator is stopped. By writing a non-zero value, the generator is started, the period is defined by the written value in milliseconds (up to 254 ms). By writing a zero value, the generator is stopped again.

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 IRC counters circuits

7.1 Introduction

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

Registers can be divided into a group common to all IRC counters

IRCCNTEnReg register intended for enabling counting or resetting by external signals

IRCCNTCtrlReg register intended for setting the counter value, or capturing the current value into the registers

and a group of registers implemented separately for each counter (the registers have names IRCCNT0..., IRCCNT1..., etc.)

IRCCNTxSetReg registers intended for data written to the "xth" counter

IRCCNTxRngReg registers intended for configuring the counting range of the "xth" counter

IRCCNTxStrReg registers intended for capturing data from the "xth" counter

IRCCNTxCWReg registers intended for configuring the "xth" counter

IRCCNTxStatReg registers intended for status information of the "xth" counter

Note: *The contents of all registers listed above except of IRCCNTxRngReg are set to zero after power-up or reset (including the reset triggered by the CardResetReg register). The value of all IRCCNTxRngReg registers is set to the maximum value, i.e. FFFFFFFF$_H$.*

7.2 IRC counter circuit functionality

The following paragraphs will describe typical counter program operations.

Counter configuration (described for counter IRCCNT0)

The IRCCNT0CWReg and IRCCNT0RngReg registers are used for configuration, allowing you to set the counter mode and counting range. Subsequently, the counter value can be set using IRCCNT0SetReg (or also IRCCNTCtrlReg).

It is recommended to check the status of the IRCCNT0StatReg register (error flag) and, if necessary, reset it using the IRCCNT0CWReg register.

Starting counters (described for counter IRCCNT0)

The IRCCNTEnReg register is used to start and stop counters (i.e. respond to input signals).

Software reading of counters (described for counter IRCCNT0)

Reading the counters takes place in two phases; in the first phase, the value of the counter (or selected counters) is captured into the buffer registers using IRCNCNTCtrllReg and in the second phase, the captured value is read using IRCCNT0StrReg.

Note: *Capturing the counter value with an external signal and stream reading using a FIFO is being prepared for future versions of the FPGA firmware.*

7.3 Differences in PCT-83xx card registers

The PCT-83xx cards differ only in the number of IRC counters or the number of SSI interfaces.

interface	PCT-8303 (.../LP)	PCT-8306 (.../LP)	PCT-8363 (.../LP)	PCT-8360 (.../LP)
number of IRC counters	3 (CNT0÷CNT2)	6 (CNT0÷CNT5)	3 (CNT0÷CNT2)	0
number of SSI interfaces	0	0	6 (SSI0÷SSI5)	6 (SSI0÷SSI5)

The following paragraphs will describe the full register structure for six IRC counters, with the understanding that for cards with three or no IRC counters, the registers, or the corresponding bits in the common registers, are not implemented.

7.4 Register IRCCNTCtrlReg [WR]

This register is intended to software capture of the current IRC counter value into the capture registers and also for setting the counter value according to preset values.

All bits can be used simultaneously in any combination, the counters allow simultaneous capturing and setting.

D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
RSRV	RSRV	STR_IRC5	STR_IRC4	STR_IRC3	STR_IRC2	STR_IRC1	STR_IRC0
D15	**D14**	**D13**	**D12**	**D11**	**D10**	**D9**	**D8**
RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV
D23	**D22**	**D21**	**D20**	**D19**	**D18**	**D17**	**D16**
RSRV	RSRV	SET_IRC5	SET_IRC4	SET_IRC3	SET_IRC2	SET_IRC1	SET_IRC0
D31	**D30**	**D29**	**D28**	**D27**	**D26**	**D25**	**D24**
RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV

STR_IRC0 stores the current value of the IRCCNT0 counter in the IRCCNT0StrReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_IRC1 stores the current value of the IRCCNT1 counter in the IRCCNT1StrReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_IRC2 stores the current value of the IRCCNT2 counter in the IRCCNT2StrReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_IRC3 stores the current value of the IRCCNT3 counter in the IRCCNT3StrReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_IRC4 stores the current value of the IRCCNT4 counter in the IRCCNT4StrReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_IRC5 stores the current value of the IRCCNT5 counter in the IRCCNT5StrReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

SET_IRC0 stores the current value of the IRCCNT0SetReg preset register in the IRCCNT0 counter
0 idle write, the state of the counter is not modified
1 data transferred to the counter (subsequent writing 0 is not required)

SET_IRC1 stores the current value of the IRCCNT1SetReg preset register in the IRCCNT1 counter
0 idle write, the state of the counter is not modified
1 data transferred to the counter (subsequent writing 0 is not required)

SET_IRC2 stores the current value of the IRCCNT2SetReg preset register in the IRCCNT2 counter
0 idle write, the state of the counter is not modified
1 data transferred to the counter (subsequent writing 0 is not required)

SET_IRC3 stores the current value of the IRCCNT3SetReg preset register in the IRCCNT3 counter
0 idle write, the state of the counter is not modified
1 data transferred to the counter (subsequent writing 0 is not required)

SET_IRC4 stores the current value of the IRCCNT4SetReg preset register in the IRCCNT4 counter
0 idle write, the state of the counter is not modified
1 data transferred to the counter (subsequent writing 0 is not required)

SET_IRC5 stores the current value of the IRCCNT5SetReg preset register in the IRCCNT5 counter
0 idle write, the state of the counter is not modified
1 data transferred to the counter (subsequent writing 0 is not required)

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

Note: *The functions controlled by the STR_IRCx bits can alternatively be controlled using the SSICtrlReg register described in the chapter dedicated to the description of the SSI interface registers.*

7.5 Register IRCCNTEnReg [WR]

This register is intended to start and stop counters, or to enable resetting by an external signal.

D7	D6	D5	D4	D3	D2	D1	D0
RSRV	RSRV	EN_AB5	EN_AB4	EN_AB3	EN_AB2	EN_AB1	EN_AB0
D15	**D14**	**D13**	**D12**	**D11**	**D10**	**D9**	**D8**
RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV
D23	**D22**	**D21**	**D20**	**D19**	**D18**	**D17**	**D16**
RSRV	RSRV	EN_R5	EN_R4	EN_R3	EN_R2	EN_R1	EN_R0
D31	**D30**	**D29**	**D28**	**D27**	**D26**	**D25**	**D24**
RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV

EN_AB0 enables IRCCNT0 counting
0 the counter is stopped
1 the counter processes signals IRCCNT0_A and IRCCNT0_B

EN_AB1 enables IRCCNT1 counting
0 the counter is stopped
1 the counter processes signals IRCCNT1_A and IRCCNT1_B

EN_AB2 enables IRCCNT2 counting
0 the counter is stopped
1 the counter processes signals IRCCNT2_A and IRCCNT2_B

EN_AB3 enables IRCCNT3 counting
0 the counter is stopped
1 the counter processes signals IRCCNT3_A and IRCCNT3_B

EN_AB4 enables IRCCNT4 counting
0 the counter is stopped
1 the counter processes signals IRCCNT4_A and IRCCNT4_B

EN_AB5 enables IRCCNT5 counting
0 the counter is stopped
1 the counter processes signals IRCCNT5_A and IRCCNT5_B

EN_R0 enables IRCCNT0 setting to zero value
0 the counter ignores IRCCNT0_R signal
1 the counter processes signal IRCCNT0_R (active level is configurable by IRCCNT0CWReg)

EN_R1 enables IRCCNT1 setting to zero value
0 the counter ignores IRCCNT1_R signal
1 the counter processes signal IRCCNT1_R (active level is configurable by IRCCNT1CWReg)

EN_R2 enables IRCCNT0 setting to zero value
0 the counter ignores IRCCNT2_R signal
1 the counter processes signal IRCCNT2_R (active level is configurable by IRCCNT2CWReg)

EN_R3 enables IRCCNT3 setting to zero value
0 the counter ignores IRCCNT3_R signal
1 the counter processes signal IRCCNT3_R (active level is configurable by IRCCNT3CWReg)

EN_R4 enables IRCCNT0 setting to zero value
0 the counter ignores IRCCNT4_R signal
1 the counter processes signal IRCCNT4_R (active level is configurable by IRCCNT4CWReg)

EN_R5 enables IRCCNT5 setting to zero value
0 the counter ignores IRCCNT5_R signal
1 the counter processes signal IRCCNT5_R (active level is configurable by IRCCNT5CWReg)

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

7.6 Register IRCCNT0SetReg [WR]

This register works as a 32-bit buffer data register for writing data to the IRCCNT0 counter; data is transferred to the IRCCNT0 counter using the IRCNCNTCtrReg register.

If a incorrect value outside the range 0÷IRCCNT0RngReg is written to the IRCCNT0 counter, the counter operates in the full 32-bit range until the counter value enters the selected range 0÷IRCCNT0RngReg.

Note: *Functionally identical registers are implemented for counters IRCCNT1 to IRCCNT5.*

7.7 Register IRCCNT0RngReg [WR]

This 32-bit register is intended to setting the counting range of the IRCCNT0 counter, valid values are 1 to 4,294,967,295 (i.e. the full range of 32-bit values) and the IRCCNT0 counter operates in the range 0÷IRCCNT0RngReg.

If such a value is written to IRCCNT0RngReg that the current state of the IRCCNT0 counter is outside the range 0÷IRCCNT0RngReg, the counter operates in the full 32-bit range until the counter value enters the selected range 0÷IRCCNT0RngReg (by processing input signals or programming a value).

Note: *Functionally identical registers are implemented for counters IRCCNT1 to IRCCNT5.*

7.8 Register IRCCNT0StrReg [RD]

This register works as a 32-bit buffer data register for software reading of IRCCNT0 counter. Current counter value is stored into IRCCNT0StrReg capture register using the IRCNCNTCtrlReg register.

Note: *Functionally identical registers are implemented for counters IRCCNT1 to IRCCNT5.*

7.9 Register IRCCNT0CWReg [WR]

This register is intended to configuring the IRCCNT0 counter.

D31÷D8	D7	D6	D5	D4	D3	D2	D1	D0
RSRV-32	RSRV	MODE			ERR	RSRV	LPF	R_CFG

R_CFG configuration of the active IRCCNT0_R signal level
- 0 the counter is set to zero value by the L level of the input signal
- 1 the counter is set to zero value by the H level of the input signal

LPF turning on/off the low-pass filter of the encoder input signals
- 0 filter is inactive
- 1 filter is activated

ERR resetting the ERR flag in IRCCNT0StatReg
- 0 idle write, flag status remains without change
- 1 the flag is reset (subsequent writing 0 is not required)

MODE configuration of the counter operating mode (described in detail in the User's Guide)
- 000 quadrature mode X1
- 001 quadrature mode X2
- 010 quadrature mode X4
- 011 reserved
- 100 "up/down" mode
- 101 "count/dir" mode
- 110 "count/gate" mode
- 111 reserved

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

RSRV-32 reserved (to ensure forward compatibility it is recommended to write 0)
although only eight bits of register are significant, it is necessary to access the register with 32-bit data

Note: *Functionally identical registers are implemented for counters IRCCNT1 to IRCCNT5.*

7.10 Register IRCCNT0StatReg [RD]

This register is intended to read the status flags of the IRCCNT0 counter.

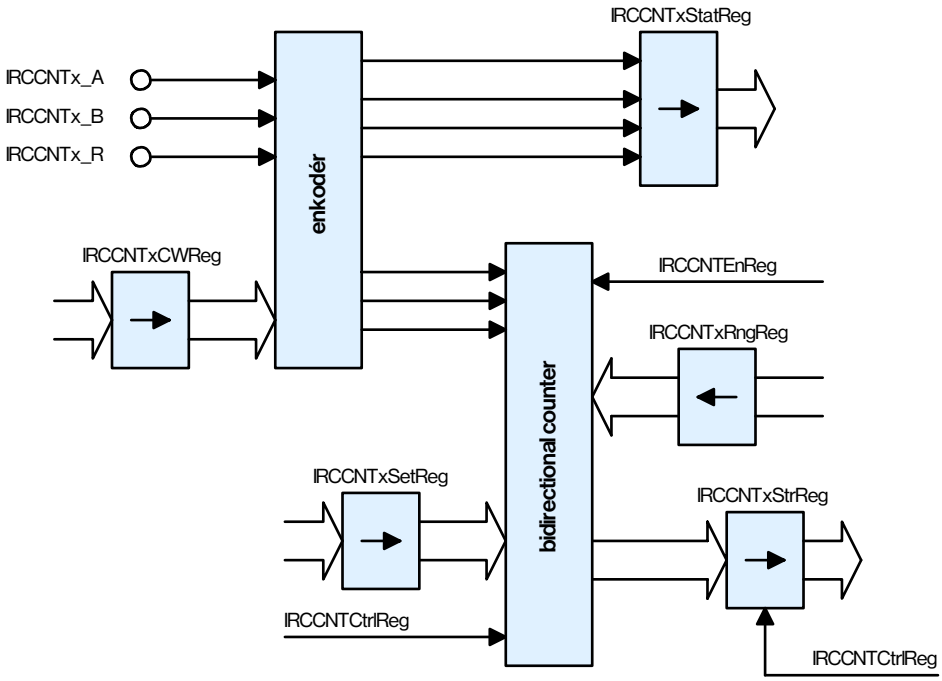
D31÷D8	D7	D6	D5	D4	D3	D2	D1	D0
RSRV-32	RSRV	RSRV	RSRV	RSRV	ERR	IRCCNT0_R	IRCCNT0_B	IRCCNT0_A

IRCCNT0_A current state of the IRCCNT0_A signal
IRCCNT0_B current state of the IRCCNT0_B signal
IRCCNT0_R current state of the IRCCNT0_R signal
ERR error flag signaling a "skip" of the quadrature signal phase in X1, X2 and X4 modes or the detected concurrent state of the IRCCNT0_A=L and IRCCNT0_B=L signals in "up/down" mode
0 no error event has been detected since the last flag reset
1 at least one error event has been detected since the last flag reset
RSRV reserved (to ensure forward compatibility it is recommended to write 0)
RSRV-32 reserved (to ensure forward compatibility it is recommended to write 0)
although only eight bits of register are significant, it is necessary to access the register with 32-bit data

Note: *Functionally identical registers are implemented for counters IRCCNT1 to IRCCNT5.*

7.11 Register structure diagram

The figure below shows the registers related to the IRC counters for better understanding (shows one counter structure; the IRCCNTx registers are implemented separately for each counter, the IRCCNT registers are common to all counters and enable their synchronous control).



8. Registers for detecting min/max values of IRC counters

8.1 Introduction

The following sections describe the registers related to counter minimum/maximum detection circuits (see the overview in Chapter 3).

Registers can be divided into a group common to all IRC counters

IRCCNTMinMaxEnReg	register intended for enabling minimum/maximum detectors
IRCCNTMinMaxCtrlReg	register intended for capturing the current value of minimum/maximum detectors

and a group of registers implemented separately for each counter (the registers have names IRCCNT0..., IRCCNT1..., etc.)

IRCCNTxMinReg	registers intended for capturing data from minimum detector of the "xth" counter
IRCCNTxMaxReg	registers intended for capturing data from maximum detector of the "xth" counter

Note: *The contents of all registers listed above except of IRCCNTxRngReg are set to zero after power-up or reset (including the reset triggered by the CardResetReg register).*

8.2 Minimum/maximum detector functionality

Each IRC counter has an independently operating minimum value detector and a maximum value detector.

All six (PCT-8303) or twelve (PCT-8306) detectors are controlled by a pair of registers IRCNCNTMinMaxEnReg and IRCNCNTMinMaxCtrlReg.

The IRCCNTMinMaxEnReg register is used to activate or restart the detectors. As long as the corresponding bit of this register is set to 0, the detector copies the current IRC counter value to the detector's internal working register. After setting the corresponding bit of the IRCCNTMinMaxEnReg register to the value 1, the detector starts comparing the current IRC counter value with internal working register value and continuously updates the value of the internal working register according to the IRC counter changes (the maximum detector updates its internal working register with each higher counter value, while the minimum detector updates its internal working register with each lower counter value).

Six or twelve capture registers controlled by the IRCCNTMinMaxCtrlReg register are intended for reading the current values of the detectors.

The min/max detectors do not support interrupts functionality and are not connected to other circuits of the card.

8.3 Differences in PCT-83xx card registers

The PCT-83xx cards differ only in the number of counters or the number of SSI interfaces.

interface	PCT-8303 (.../LP)	PCT-8306 (.../LP)	PCT-8363 (.../LP)	PCT-8360 (.../LP)
number of IRC counters	3 (CNT0÷CNT2)	6 (CNT0÷CNT5)	3 (CNT0÷CNT2)	0
number of SSI interfaces	0	0	6 (SSI0÷SSI5)	6 (SSI0÷SSI5)

The following paragraphs will describe the full register structure for six IRC counters, with the understanding that for cards with three or no IRC counters, the registers, or the corresponding bits in the common registers, are not implemented.

8.4 Register IRCCNTMinMaxEnReg [WR]

This register is intended to activate/deactivate of all six minimum and six maximum detectors.

D7	D6	D5	D4	D3	D2	D1	D0
RSRV	RSRV	EN_MIN5	EN_MIN4	EN_MIN3	EN_MIN2	EN_MIN1	EN_MIN0
D15	**D14**	**D13**	**D12**	**D11**	**D10**	**D9**	**D8**
RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV
D23	**D22**	**D21**	**D20**	**D19**	**D18**	**D17**	**D16**
RSRV	RSRV	EN_MAX5	EN_MAX4	EN_MAX3	EN_MAX2	EN_MAX1	EN_MAX0
D31	**D30**	**D29**	**D28**	**D27**	**D26**	**D25**	**D24**
RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV

EN_MIN0 enables/disables the IRCCNT0 counter minimum detector
0 detector functionality is stopped
1 detector functionality is running (restarted by transition 0=>1)

EN_MIN1 enables/disables the IRCCNT1 counter minimum detector
0 detector functionality is stopped
1 detector functionality is running (restarted by transition 0=>1)

EN_MIN2 enables/disables the IRCCNT2 counter minimum detector
0 detector functionality is stopped
1 detector functionality is running (restarted by transition 0=>1)

EN_MIN3 enables/disables the IRCCNT3 counter minimum detector
0 detector functionality is stopped
1 detector functionality is running (restarted by transition 0=>1)

EN_MIN4 enables/disables the IRCCNT4 counter minimum detector
0 detector functionality is stopped
1 detector functionality is running (restarted by transition 0=>1)

EN_MIN5 enables/disables the IRCCNT5 counter minimum detector
0 detector functionality is stopped
1 detector functionality is running (restarted by transition 0=>1)

EN_MAX0 enables/disables the IRCCNT0 counter maximum detector
0 detector functionality is stopped
1 detector functionality is running (restarted by transition 0=>1)

EN_MAX1 enables/disables the IRCCNT1 counter maximum detector
0 detector functionality is stopped
1 detector functionality is running (restarted by transition 0=>1)

EN_MAX2 enables/disables the IRCCNT2 counter maximum detector
0 detector functionality is stopped
1 detector functionality is running (restarted by transition 0=>1)

EN_MAX3 enables/disables the IRCCNT3 counter maximum detector
0 detector functionality is stopped
1 detector functionality is running (restarted by transition 0=>1)

EN_MAX4 enables/disables the IRCCNT4 counter maximum detector
0 detector functionality is stopped
1 detector functionality is running (restarted by transition 0=>1)

EN_MAX5 enables/disables the IRCCNT5 counter maximum detector
0 detector functionality is stopped
1 detector functionality is running (restarted by transition 0=>1)

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

8.5 Register IRCCNTMinMaxCtrlReg [WR]

This register is intended to software capture of the current values of internal working registers of minimum or maximum detectors into capture registers. All bits can be used simultaneously in any combination.

D7	D6	D5	D4	D3	D2	D1	D0
RSRV	RSRV	STR_MIN5	STR_MIN4	STR_MIN3	STR_MIN2	STR_MIN1	STR_MIN0
D15	**D14**	**D13**	**D12**	**D11**	**D10**	**D9**	**D8**
RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV
D23	**D22**	**D21**	**D20**	**D19**	**D18**	**D17**	**D16**
RSRV	RSRV	STR_MAX5	STR_MAX4	STR_MAX3	STR_MAX2	STR_MAX1	STR_MAX0
D31	**D30**	**D29**	**D28**	**D27**	**D26**	**D25**	**D24**
RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV

STR_MIN0 stores the current value of the IRCCNT0 minimum detector in the IRCCNT0MinReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_MIN1 stores the current value of the IRCCNT1 minimum detector in the IRCCNT1MinReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_MIN2 stores the current value of the IRCCNT2 minimum detector in the IRCCNT2MinReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_MIN3 stores the current value of the IRCCNT3 minimum detector in the IRCCNT3MinReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_MIN4 stores the current value of the IRCCNT4 minimum detector in the IRCCNT4MinReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_MIN5 stores the current value of the IRCCNT5 minimum detector in the IRCCNT5MinReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_MAX0 stores the current value of the IRCCNT0 maximum detector in the IRCCNT0MaxReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_MAX1 stores the current value of the IRCCNT1 maximum detector in the IRCCNT1MaxReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_MAX2 stores the current value of the IRCCNT2 maximum detector in the IRCCNT2MaxReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_MAX3 stores the current value of the IRCCNT3 maximum detector in the IRCCNT3MaxReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_MAX4 stores the current value of the IRCCNT4 maximum detector in the IRCCNT4MaxReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_MAX5 stores the current value of the IRCCNT5 maximum detector in the IRCCNT5MaxReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

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

8.6 Register IRCCNT0MinReg [RD]

This register works as a 32-bit buffer data register for software reading of the IRCCNT0 counter minimum detector data, the value is captured using the IRCCNTMinMaxCtrlReg register.

Note: Functionally identical registers are implemented for minimum detectors of counters IRCCNT1 to IRCCNT5.

8.7 Register IRCCNT0MaxReg [RD]

This register works as a 32-bit buffer data register for software reading of the IRCCNT0 counter maximum detector data, the value is captured using the IRCCNTMinMaxCtrlReg register.

Note: Functionally identical registers are implemented for maximum detectors of counters IRCCNT1 to IRCCNT5.

9. Registers intended for SSI interfaces

9.1 Introduction

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

Registers can be divided into a group common to all SSI interfaces

SSICfgReg	register intended for timing configuration of all SSI interfaces
SSICtrlReg	register intended for capturing the last value read from the SSI interface into the registers (or also for capturing the current value of IRC counters)

and a group of registers implemented separately for each SSI interface (the registers have names SSI0..., SSI1..., etc.)

SSIxCfgReg	registers intended for configuring the "xth" SSI interface
SSIxStrReg	registers intended for capturing data from the "xth" SSI interface

Note: *The contents of all registers listed above are set to zero after power-up or reset (including the reset triggered by the CardResetReg register).*

9.2 SSI interfaces circuit functionality

The following paragraphs will describe typical SSI interfaces program operations.

Configuration of SSI interfaces

Each SSI interface has one register SSIxCfgReg intended to define the transmitted data bit length (in the range of 1÷32 bits) and the working data code (direct binary or Gray code).

Configuration of SSI interfaces controller

The SSI interface controller has one register SSICfgReg intended to define the operating frequency of the CLK_SSIx signals (in the range of 100 kHz ÷ 1 MHz) and the data loading period (in the range of 10÷256 CLK signal periods). This setting is intended to all SSI interfaces.

By configuring a valid frequency starts the operation of all SSI interfaces.

Software reading of SSI interface controller data

Data transfer from sensors occurs automatically with a period selected by the SSICfgReg register and sensor data are stored in the internal data registers of the SSI controller. The current contents of the internal data registers can be captured to the SSIxStrReg registers by a command written to the SSICtrlReg register.

Note: *Capturing the counter value with an external signal and stream reading using a FIFO is being prepared for future versions of the FPGA firmware.*

9.3 Differences in PCT-83xx card registers

The PCT-83xx cards differ only in the number of counters or the number of SSI interfaces.

interface	PCT-8303 (.../LP)	PCT-8306 (.../LP)	PCT-8363 (.../LP)	PCT-8360 (.../LP)
number of IRC counters	3 (CNT0÷CNT2)	6 (CNT0÷CNT5)	3 (CNT0÷CNT2)	0
number of SSI interfaces	0	0	6 (SSI0÷SSI5)	6 (SSI0÷SSI5)

The registers described in the following paragraphs (except SSICtrlReg) are implemented only on the PCT-8360 and PCT-8363 cards; the SSICtrlReg register is implemented on all four PCT-83xx types described in this manual.

9.4 Register SSICtrlReg [WR]

This register is dediacated for software capture current value of internal SSI sensors data registers data (and also capture of the current IRC counter value) to registers intended for software reading.

All bits can be used simultaneously in any combination, the counters allow simultaneous reading and setting.

D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
RSRV	RSRV	STR_SSI5	STR_SSI4	STR_SSI3	STR_SSI2	STR_SSI1	STR_SSI0
D15	**D14**	**D13**	**D12**	**D11**	**D10**	**D9**	**D8**
RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV
D23	**D22**	**D21**	**D20**	**D19**	**D18**	**D17**	**D16**
RSRV	RSRV	STR_IRC5	STR_IRC4	STR_IRC3	STR_IRC2	STR_IRC1	STR_IRC0
D31	**D30**	**D29**	**D28**	**D27**	**D26**	**D25**	**D24**
RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV

STR_SSI0 stores the current value of internal SSI sensors data register in the SSI0StrReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_SSI1 stores the current value of internal SSI sensors data register in the SSI1StrReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_SSI2 stores the current value of internal SSI sensors data register in the SSI2StrReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_SSI3 stores the current value of internal SSI sensors data register in the SSI3StrReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_SSI4 stores the current value of internal SSI sensors data register in the SSI4StrReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_SSI5 stores the current value of internal SSI sensors data register in the SSI5StrReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_IRC0 stores the current value of the IRCCNT0 counter in the IRCCNT0StrReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_IRC1 stores the current value of the IRCCNT1 counter in the IRCCNT1StrReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_IRC2 stores the current value of the IRCCNT2 counter in the IRCCNT2StrReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_IRC3 stores the current value of the IRCCNT3 counter in the IRCCNT3StrReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_IRC4 stores the current value of the IRCCNT4 counter in the IRCCNT4StrReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

STR_IRC5 stores the current value of the IRCCNT5 counter in the IRCCNT5StrReg capture register
0 idle write, the state of the capture register is not modified
1 data transferred to the capture register (subsequent writing 0 is not required)

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

Note: *The functions controlled by the STR_IRCx bits are intended for alternative control of IRC counters (see description in the chapter dedicated to IRC counter registers).*
The SSICtrlReg register is also implemented in PCT-8303/8306 cards, i.e. types without SSI interfaces.

9.5 Register SSICfgReg [WR]

This register is intended to configure the timing of all SSI interfaces.

D7	D6	D5	D4	D3	D2	D1	D0
RSRV	RSRV	RSRV	RSRV	CLK_FRQ			
D15	**D14**	**D13**	**D12**	**D11**	**D10**	**D9**	**D8**
SSI_PER							
D23	**D22**	**D21**	**D20**	**D19**	**D18**	**D17**	**D16**
RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV
D31	**D30**	**D29**	**D28**	**D27**	**D26**	**D25**	**D24**
RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV

CLK_FRQ defines the frequency of the CLK_SSIx signals (shared by all SSI interfaces)
- 0 SSI interface controller stopped, all CLK_SSIx signals set permanently to H level
- 1 CLK_SSIx signals operate at a frequency of 100 kHz
- 2 CLK_SSIx signals operate at a frequency of 200 kHz
- ... ...
- 10 CLK_SSIx signals operate at a frequency of 1 MHz
- 11 reserved
- ... ...
- 15 reserved

SSI_PER defines the data loading period (shared by all SSI interfaces)
- 0÷8 reserved (identical function as for value 9)
- 9 the data loading period is set to 10 periods of the CLK_SSIx signal (see CLK_FRQ)
- ... ...
- 255 the data loading period is set to 256 periods of the CLK_SSIx signal (see CLK_FRQ)

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

9.6 Register SSI0CfgReg [WR]

This register is intended to configure the data bit length and the working data code.

D7	D6	D5	D4	D3	D2	D1	D0
RSRV	RSRV	RSRV	DATA_Lenght				
D15	**D14**	**D13**	**D12**	**D11**	**D10**	**D9**	**D8**
RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	DATA_Code	
D23	**D22**	**D21**	**D20**	**D19**	**D18**	**D17**	**D16**
RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV
D31	**D30**	**D29**	**D28**	**D27**	**D26**	**D25**	**D24**
RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV	RSRV

DATA_Lenght defines the number of bits transmitted by the SSI interface
- 0 number of bits = 1 (this value should also be set for unused SSI interfaces)
- ... ...
- 31 number of bits = 32

DATA_Code defines the code of data transmitted by the SSI interface
- 0 direct binary code
- 1 Gray code
- 2 a 3 reserved

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

Note: *Functionally identical registers are implemented for interfaces SSI1 to SSI5.*

9.7 Register SSI0StrReg [RD]

This register is intended for storing value captured from internal SSI sensor data register (sensor connected to interface 0).

Note: *Functionally identical registers are implemented for interfaces SSI1 to SSI5.*

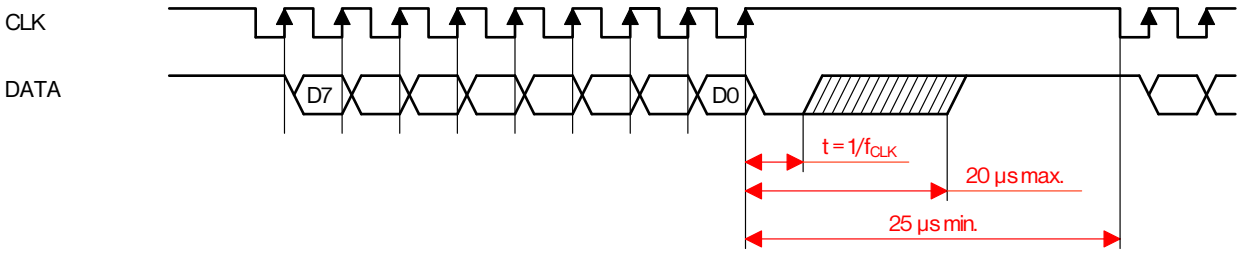
9.8 Description of the SSI interface controller operation and register structure

The figures below illustrate the principles of SSI interface transfers and describe the functions of registers related to the SSI interface for better understanding.

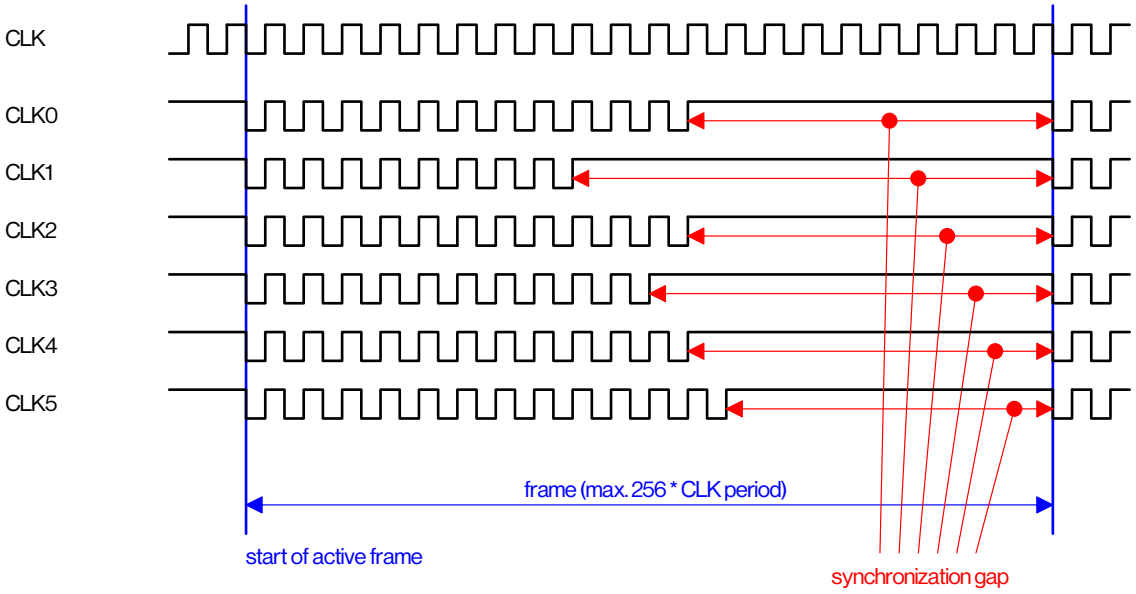
SSI interface signals

The SSI interface consists (in addition to the supply voltage) of two signals named CLK and DATA; both are transmitted by a differential pair of signals at RS-422 voltage levels.

The CLK signal is generated by the SSI master (i.e. PCT-836x card), the DATA signal is generated by the sensor connected to the SSI master controller. A gap between groups of CLK pulses with a width of at least 25 µs is used for synchronization; the sensor detects this gap and, after the next sequence of CLK pulses begins, transmits data starting with the highest bit.



Generation of CLK signals by the card's internal circuits



Bits D3÷D0 of the SSICfgReg register are intended for selecting the frequency of all CLKx signals; internal circuits works as a divider creating a signal with a frequency of 100 kHz to 1 MHz (marked as CLK at the very top in the figure).

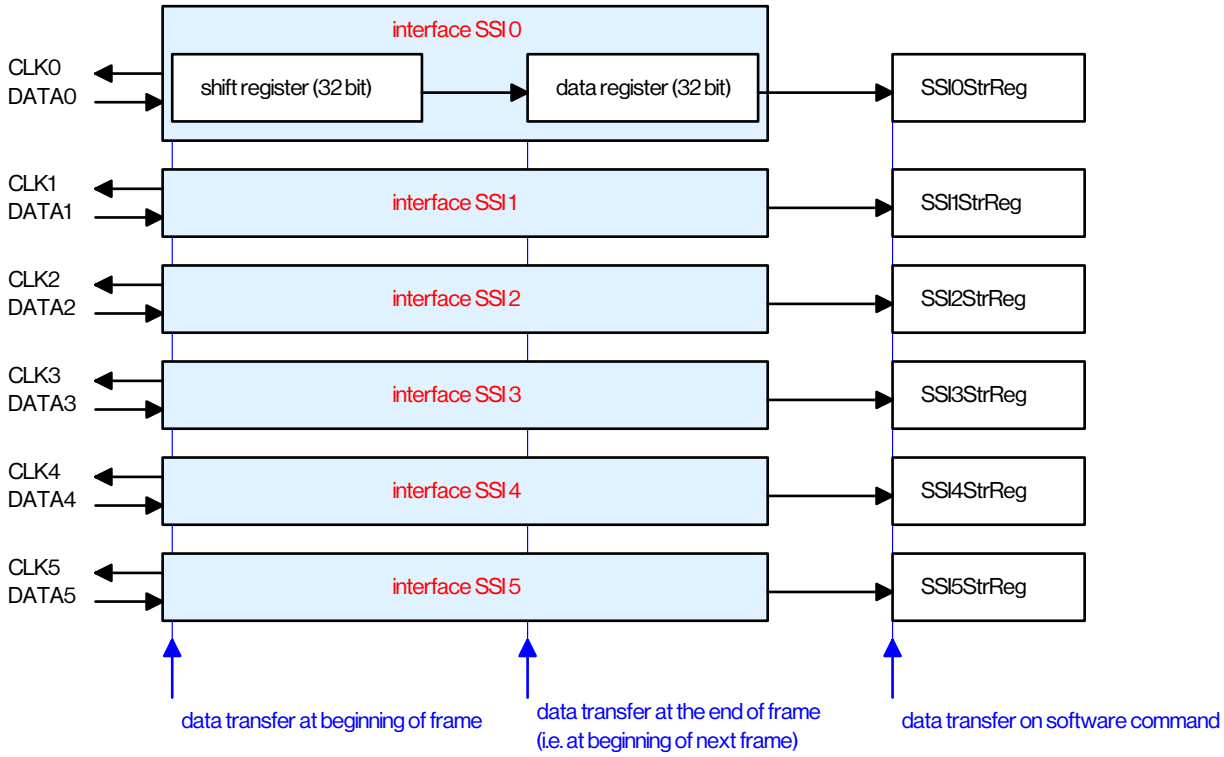
The SSI controller frame time (i.e. the period with which data is read from the SSI interface) is defined by bits D15÷D8 of the SSICfgReg register in the range of 10 to 256 periods of the CLK signal (marked by two blue lines in the figure).

The number of bits read from the SSI interface can be configured for each interface by bits D4÷D0 of the SSIxCfgReg registers in the range of 1 to 32 bits (actually, 2÷33 CLK pulses are generated in accordance with the SSI specification). The six CLKx signals for different SSICfgReg and SSIxCfgReg settings are shown in the figure.

To correctly configure the SSI interface, it is necessary to ensure that for each CLK signal (or for the CLK signal of the SSI interface, which is configured for the highest number of bits of all) a synchronization gap of at least 25 µs is guaranteed; therefore, it is necessary to set a sufficient frame length with respect to the longest bit length of the transmission and the selected frequency (setting a too long frame time or a low CLK frequency unnecessarily prolongs the value update time).

Register structure

The figure below shows the data flows through the SSI interface.



Data from the SSI sensor is first stored in a shift register by the SSI controller (individual bits are written starting from the second rising edge of the CLK signal following the synchronization gap).

At the end of a frame (corresponding to the start of a new frame), the data from the shift registers are transferred to the internal data registers of the SSI controller.; due to the configurable data length, the lowest bits of the data register are used and unused bits remain at zero value. During transfer from the shift registers SSI sensor values are converted from Gray code to direct binary code, if Gray code mode is configured.

As can be seen from the previous paragraphs, the data of the internal data registers of SSI controllers are periodically updated; the SSIxStrReg registers are used for reading, their content is updated on command from the software via the SSICtrlReg register.

10. Diagnostic registers (common to all card types)

10.1 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

10.2 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.*

10.3 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.

10.4 Register CardSerNrReg [RD]

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

10.5 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.

10.6 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.*

10.7 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.*

11. Registers in address spaces BAR1 a BAR2

11.1 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.*

11.2 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.*

11.3 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.*