DAQ PCI/PCIe card's Windows system driver

Programmer's Guide

Document history		
date	version	changes
29.2.2016	02.2016/B	the initial release (translation from Czech, driver version 3.12)
30.3.2021	03.2021	New features of the library tedia_ep4gxa.dll (driver version 3.18)… - new functions for accessing registers with 64-bit data - new functions for reading Serial Number and CardID Number (libraries tedia_ox9162.dll and tedia_ox952.dll without changes)
12.5.2022	05.2022	chapter D4.1 changes (driver version 3.20 and 3.22)

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.

Table of Contents

Appendix C, tedia_ep4gxa.dll library

Appendix D, list of supported cards & driver's history

1. General information

1.1 Introduction

This programmer's guide is dedicated to describing the system drivers supplied with TEDIA cards in case the user needs to create their own program by directly controlling the card's registers. It is not intended for a regular card user who only needs to install and use the system driver with already created programs, or to program an application using one of the application drivers (eg. TEDIA_DAQ01, drivers for the Control Web development system, ...).

This programmer's guide contains...

- description of the system driver, file structure, etc.;
- description of API (interface of library);
- how to create your own program or own application driver using the system driver.

What this guide does not contain...

- description of driver installation (refer to the separate manual dedicated to installation only);
- description of application drivers (eg TEDIA_DAQ01) running over the system driver.

To use the driver, it is necessary to know the card's register structure, usually described in the programmer's guide of the selected card. A description of the registry structure is available for all types of DAQ PCI and PCIe TEDIA cards.

1.2 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. General information and terminology

2.1 Introduction

The following paragraphs contain a general description of PCI/PCIe TEDIA cards and description of the terms used in the following chapters of this manual.

2.2 Terminology

A number of terms are used in the description of system driver, the meaning of which may not be completely clear to every user. The basic terms are described in the following paragraphs.

system driver	is software containing basic card's control code in the Windows kernel and uses a common DLL library as an interface for common programs (or application drivers)
application driver	is software that uses the functions of the system driver (towards the card) and in the opposite direction provides specific functions and interfaces for various development tools and systems; no part of application driver runs in the Windows kernel
PCI-SIG	PCI Special Interest Group; organization for PCI and PCI Express bus standardization
PCI VID	PCI Vendor ID; is a 16-bit long identification number assigned to the PCI-SIG member; (TEDIA PCI VID = 1760_H)
PCI DID	PCI Device ID, is a 16-bit number assigned by the manufacturer (or PCI VID holder) to each type of card
... VID + DID	set of these two values 100% identifies the type of card
PCI BUS	the PCI bus number in the computer
PCI SLOT	the slot number for the PCI/PCIe plug-in card in the computer
... BUS + SLOT	set of these two values 100% defines the location of the card in the computer and allows you to identify the card, especially if there are two or more cards of the same type in the computer
PCI function	the add-on card can contain up to eight separate functional parts, which the system recognizes and operates completely independently (simply said - one add-on card with two PCI functions behaves as if two separate cards installed in one slot)
BAR	Base Address Registr; PCI/PCIe cards contain function registers that are mapped to the I/O or MEM space (during starting, the operating system detects the card's requirements and writes the base addresses of register's blocks to the BAR registers); the card (or each PCI function) can contain several register blocks, each with its own BAR register and assigned address
BARs of TEDIA cards	cards use multiple register blocks, but only one block (BAR) is reserved for user registers; the BAR number depends on the card group (see description in the following chapter)
I/O or MEM	card registers can be mapped in I/O space (obsolete, slow, and useful only for operating systems that do not allow 32-bit memory addressing) and in MEM space; the Windows system driver uses registers mapped only in the MEM space

2.3 TEDIA cards

From the perspective of the PCI/PCIe controller all TEDIA cards can be divided into three groups.

PCI cards, 1. gen.	• equipped with OX9162 (Oxford Semiconductor) • compatible only with 5 V signal levels
PCI cards, 2. gen.	• equipped with OXuPCI952 controller (PLX Technology, formerly Oxford Semiconductor) • compatible with both 5 V and 3,3 V signal levels
PCI Express cards	• equipped with an EP4GX series FPGA (Intel, formerly Altera) • compatible with the PCI Express bus x1 Gen 1.0a/1.1

From the perspective of the Windows system driver, all TEDIA cards are serviced by a common system part of driver (running in the Windows kernel) and three separate DLL libraries creating API interfaces. The driver does not functionally differentiate among types of cards within one group (ie. the driver controls all types within one group completely unified).

3. Description of the driver and programming interface

3.1 Introduction

The following paragraphs are dedicated to a general description of the driver and the implemented functions.

3.2 Driver components

The system driver consists of the following key files...

tediaOxPCI.sys	the part running in the Windows kernel
tedia_ox9162.dll	interface for 1. generation PCI equipped with OX9162
tedia_ox952.dll	interface for 2. generation PCI equipped with OXuPCI952 (or OXmPCI952) controller
tedia_ep4gxa.dll	interface for PCI Express cards equipped with an EP4GX series FPGA

All files mentioned above are available in both 32-bit and 64-bit versions and allow 32-bit programs installed in both 32-bit and 64-bit versions of Windows, resp. 64-bit programs installed on 64-bit versions of Windows.

The installation package also contains other files, but their description is beyond the scope of this document and their meaning is not relevant to users.

Note: *The API descriptions of the three DLLs are given in the appendices to this document; the documentation of the tediaOxPCI.sys interface is not public.*

3.3 Overview of library functions

The tables below list the functions of the libraries according to their meaning, and the appendices to this manual are given a detailed description.

As will be seen from the tables, all libraries contain (with marked exceptions) identical functions differing only by prefix (analogous functions are named *OX9162_**, *OX952_** or *EP4GXA_** depending on the library).

General functions independent of installed cards	
function	**description**
OX9162_InterfaceVersion OX952_InterfaceVersion EP4GXA_InterfaceVersion	functions dedicated to determine version of the interface part of driver (ie. all DLLs)
OX9162_DriverVersion OX952_DriverVersion EP4GXA_DriverVersion	functions dedicated to determine version of the system part of driver (ie. tediaOxPCI.sys)
OX9162_CardsInSystem OX952_CardsInSystem EP4GXA_CardsInSystem	functions dedicated to determine the number of cards currently available to the driver (separately for each group of cards)

As can be seen from the table above, the API contains functions that allow to determine the version of the interface (ie. DLL library) and the system part of the driver (ie. tediaOxPCI.sys).

The third function allows to determine the number of cards available to the driver. All currently available cards are sorted into three tables (one for each of the DLLs) and the function returns the number from zero (no card available) above.

The functions accessing the card use its location in the table as the identifier of the card; this index takes values from zero (the first card in the table) up.

Function to determine the properties of cards (usable even before opening the connection to the card)	
function	**description**
OX9162_Card_DID OX952_Card_DID EP4GXA_Card_DID	functions dedicated to determine the PCI Device ID of the selected card (in conjunction with TEDIA VID defines the type of card)
OX9162_Card_BUS OX952_Card_BUS EP4GXA_Card_BUS	functions dedicated to determine the PCI bus number where the selected card is installed
OX9162_Card_SLOT OX952_Card_SLOT EP4GXA_Card_SLOT	functions dedicated to determine the PCI slot number where the selected card is installed
EP4GXA_Card_ID	function dedicated to determine the CardID value of the selected card (user configurable by DIP switch)
EP4GXA_Card_SerNr	function dedicated to determine the serial number of the selected card

The second group of functions is dedicated to determine the properties of the card, resp. location on the computer.

All three functions use the card 's index in the table as an identifier (see the description of the functions ****_CardsInSystem* in the previous paragraph)and provide information about...

PCI DID — in conjunction with TEDIA VID defines the type of card
(the driver does not support third-party PCI VID cards)

BUS + SLOT — a pair of constants that allow to identify a card by its location in your computer
(needed especially when there are two or more cards of the same type installed in computer)

Functions for opening/closing the connection to the card	
function	**description**
OX9162_Open OX952_Open EP4GXA_Open	functions dedicated to open an exclusive connection to the selected card (ie. another program cannot open the connection using ****_Open*, nor ****_OpenMulti* functions)
OX9162_OpenMulti OX952_OpenMulti EP4GXA_OpenMulti	functions dedicated to open an non-exclusive connection to the selected card (ie. another program can open the connection using ****_OpenMulti* functions, but cannot open the connection using ****_Open* functions)
OX9162_Card_OpenCount OX952_Card_OpenCount EP4GXA_Card_OpenCount	functions dedicated to determine the current number of open connection to the selected using ****_OpenMulti* functions
OX9162_Close OX952_Close EP4GXA_Close	functions dedicated to close the connection to the card

The third group of functions is intended for opening and closing connections to the card. The table also contain a related auxiliary functions allowing to determine the current number of open connections to the selected card.

Libraries contain two functions for opening a connection; basic functions ****_Open* open an exclusive connection to the card, ie. block simultaneous opening by another program.

Alternative functions ****_OpenMulti* allow to open multiple (non-exclusive) connections to the card and access the card to more programs at the same time (all programs must use ****_OpenMulti*). The ****_Card_OpenCount* functions allow to determine the number of programs that are currently using the selected card.

Functions to access to the card registers (open connection to the card required)	
function	**description**
OX9162_BAR4_BaseAdr (OX952_BAR05_BaseAdr) OX952_BAR11_BaseAdr EP4GXA_BAR0_BaseAdr	functions dedicated to determine the base address of the memory space block where are the user registers of the card mapped (*OX952_BAR05_BaseAdr* see a note in the text below the table)
OX9162_MemRead OX952_MemRead EP4GXA_ReadReg8 EP4GXA_ReadReg32 EP4GXA_ReadReg64 *** EP4GXA_ReadReg2x32 ***	OX9162 and OX952 API interfaces offer functions for reading or writing 8-bit registers (address range is not limited at all); EP4GXA API interface offers functions with different parameters for reading or writing of one 8-bit or 32-bit registers, respectively two 32-bit registers at once (unlike OX9162 and OX952, a valid address is limited only to the card allocated register space) *** new function available in version 3.18 and higher
OX9162_MemWrite OX952_MemWrite EP4GXA_WriteReg8 EP4GXA_WriteReg32 EP4GXA_WriteReg64 *** EP4GXA_WriteReg2x32 ***	

The fourth group of functions is intended for access to the card registers.

The ****_BAR*_BaseAdr* functions allow to determine the base address (ie. starting address) of a memory block used by the user registers; the address wide of 32-bit or 64-bit depends on the program (resp. on the version of the attached DLL). The mapping of user registers of a specific card is described in the card user manual as an offset from the address passed by the functions ****_BAR*_BaseAdr*.

Note: *The OXmPCI952/OXuPCI952 driver provides, in addition to BAR1 of PCI function 1 (marked as BAR11) mapping user function registers, also BAR5 of PCI function 0 (marked as BAR05) containing registers of serial port used for service purposes. The basic function of the service interface is the possibility of user-friendly updating the card firmware, but it also allows to determine the unique serial number of the card.*

Interrupt handling functions (open connection to the card required)	
function	**description**
OX9162_IRQ_Wait OX952_IRQ_Wait EP4GXA_IRQ_Wait	functions stop the thread until the interrupt event occurs or timeout period expires
OX9162_IRQ_SetTimeout OX952_IRQ_SetTimeout EP4GXA_IRQ_SetTimeout	functions set the timeout period used by the ****_IRQ_Wait* function
OX9162_IRQ_Count OX952_IRQ_Count EP4GXA_IRQ_Count	functions pass the number of interrupt events triggered by the card

The last group of functions creates a simple mechanism for capturing system interruptions by the card.

The main function is ****_IRQ_Wait*, which suspends the thread until a system interruption or timeout expires. The program periodically calls this function in a separate thread and, depending on the return code, starts the interrupt event code.

The ****_IRQ_SetTimeout* function allows to set the time for which the ****_IRQ_Wait* function releases a thread if no system interruption is detected (ie. no event occurs).

The last function can be used to determine the number of interruptions caused by the card.

4. Description of using the driver

4.1 Introduction

The following paragraphs are dedicated to a general description of how to use the driver by user programs..

4.2 How to find a card and open a connection

If it is necessary to ensure simultaneous access from several programs (or processes), the function ****_OpenMulti* can be used to open a connection to the card. However, it is necessary to realize that the control rights of all processes are symmetrical and in the case of access to the same functions/registers they can cause mutual collisions. In all other cases, it is better to use the ****_Open* function to open a exclusive connection to the card, blocking the simultaneous opening of another program.

However, the actual opening of the connection to the card should always be preceded by determining which cards and on which indexes of the table are available. Each of the *tedia_ox9162.dll, tedia_ox952.dll and tedia_ep4gxa.dll* libraries creates its own table.

In the first step, it is necessary to use the ****_CardsInSystem* function to determine how many cards are available to the selected library. If the type of card with which the program is ready to work is known, it is sufficient to use the only ones from the libraries *tedia_ox9162.dll, tedia_ox952.dll or tedia_ep4gxa.dll*, in the case of various diagnostic programs it is possible to determine the status for all libraries.

In the second step, the ****_Card_DID* function helps to determine PCI Device ID for all available cards of the selected library (ie. from 0 to the value passed by the ****_CardInSystem* function reduced by 1) and select the card / cards with which the program will work. A list of supported cards and their PCI Device IDs is provided in Appendix D of this guide. In the case of multiple identical cards in the computer, the data provided by the ****_Card_BUS* and ****_Card_SLOT* functions can be used to distinguish them.

In the third step can be the connection to the card opened with one of the functions ****_Open* or ****_OpenMulti*.

4.3 How to access to user card's registers

If the connection to the PCI card is successfully opened, it is necessary to determine the starting address (ie. base address) of the register block using the *OX9162_BAR4_BaseAdr* or *OX952_BAR11_BaseAdr* function. The address wide of 32-bit or 64-bit depends on the program (resp. on the version of the attached DLL). The standard direct memory access method can be used for access to the registers, or access using the ****_MemRead* and ****_MemWrite* functions (ie. a pair of *OX9162_MemRead* and *OX9162_MemWrite*, or a pair of *OX952_MemRead* and *OX952_MemWrite*).

In the case of PCI Express cards, the *EP4GXA_BAR1_BaseAdr* function and standard direct memory access method can be used, however, it is more appropriate and safer to use *EP4GXA_ReadReg** a *EP4GXA_WriteReg** functions. Moreover, it is quite difficult to replace functions that transfer 64-bit data within a single instruction in 32-bit compiled programs.

4.4 How to use the interrupt handling mechanism

Interrupt handling can be divided into three parts...

- the first is access to the card registers to configure the interrupt source (functionality provided by user program); this step also includes disabling interrupt sources when the program terminates
- the second is to capture the interrupt event itself and transfer this information to the user program (functionality provided by the driver)
- the third is the identification of the interrupt source and the service code depending on the interrupt source (functionality provided by user program).

While the second part is unified for all card types within a group controlled by one library (solved by the ****_IRQ_Wait*, ****_IRQ_SetTimeout* and ****_IRQ_Count* functions), the first and third parts are specific to a card type and interrupt event as well.

For an overview several words about the registers of TEDIA cards related to system interrupt triggering circuits.

Most cards have a set of three registers for enabling the interrupt source (IRQCfgReg), identifying the interrupt event (IRQStatusReg), and clearing the set event flag (IRQClrReg). In addition to these registers, the cards have a fourth register that allows to globally enable or disable generating an interrupt event (INTEnReg). A detailed description can be found in the programmer's guide of the selected card.

In general, the following sequence of operations can be defined (the default is to disable all interrupt sources and cleared all identification flags, and no interrupts are captured from previous activity, see below):

- global enable of generating an interrupt event by the INTEnReg register (must remain enabled for all the time of interrupt mechanism would be used)
- enable selected interrupt sources by the IRQCfgReg register (and configuring/activating/starting the peripheral that uses this interrupt source)
- cyclic calling of the ****_IRQ_Wait* function in a separate thread; in case of return code ****_OK* (one or more simultaneously triggered interrupts) the handling routine/routines will be started, consisting of basic steps
  - identify the source of the interrupt event using the IRQStatusReg register
  - specific data processing (for example reading of measured data from FIFO memory)
  - clearing the interrupt flag by the IRQClrReg register

  Capability of generating an interrupt event with the INTEnReg register must remain enabled at all times!
- disable selected interrupt sources by the IRQCfgReg register
- clearing any possible interrupt flags by the IRQClrReg register
- global disable of generating an interrupt event by the INTEnReg register

The sequence of operations described in the previous paragraphs assumes, by default, the disabling of all interrupt sources, the clearing of all identification flags, and no interrupts in a que from previous card activity. This state can be achieved by the following procedure:

- disable all interrupt sources by the IRQCfgReg registry
- global disabling of generating an interrupt event by the INTEnReg registry
- reuse of function ****_IRQ_Wait* until return code ****_WaitTimeout*; the function is called once (no interrupts in a que from the previous activity) or twice (first return code ****_OK* and second time ****_WaitTimeout*); it is reasonable to set a short timeout; this part does not need to be performed if the connection to the card is terminated using the ****_Close*

4.5 How to close a connection

The function *** _ Close is dedicated to end the connection to the card. However, it should be preceded by:

- setting the card registers to a safe state (for example, terminating the measurement and storing data in the FIFO memory), especially the registers related to the interrupt (see the description in the previous paragraphs)
- termination of the interrupt service thread (ie. using the ****_IRQ_Wait* function), but in any case it is necessary to evaluate the return code ****_StopInThread*

After performing the ****_Close* function, it is no longer possible to access the card registers or handle interrupts; if the program is to re-access the card registers, it is necessary to reopen the connection to the card. **At this point, however, it needs to be emphasized, that a new register block address is assigned each time it is opened.**

A1.1 Function set of the tedia_ox9162.dll library version 3.00 and later

The table below provides an overview of all public functions implemented in library, a detailed description follows on next pages of this appendix.

General functions independent of installed cards	
function	**description**
OX9162_InterfaceVersion	driver interface version (ie. tedia_ox9162.dll)
OX9162_DriverVersion	system driver version (ie. tedia_OxPCI.sys)
OX9162_CardsInSystem	number of cards available to the tedia_ox9162.dll library
Function to determine the properties of cards (usable even before opening the connection to the card)	
function	**description**
OX9162_Card_DID	the PCI Device ID of the selected card (in conjunction with TEDIA VID defines the type of card)
OX9162_Card_BUS	the PCI bus number where the selected card is installed
OX9162_Card_SLOT	the PCI slot number where the selected card is installed
Functions for opening/closing the connection to the card	
function	**description**
OX9162_Open	opens an exclusive connection to the selected card (ie. another program cannot open the connection using OX9162_OpenMulti, nor OX9162_Open)
OX9162_OpenMulti	opens an non-exclusive connection to the selected card (ie. another program can open the connection using OX9162_OpenMulti, but cannot open the connection using OX9162_Open)
OX9162_Card_OpenCount	current number of open connection using OX9162_OpenMulti function
OX9162_Close	closes the connection to the card
Functions to access to the card registers (open connection to the card required)	
function	**description**
OX9162_BAR4_BaseAdr	the starting address and size of the memory block mapping the functional registers of the card (BAR4); necessary for direct register access
OX9162_MemRead	function for reading the card register (8-bit data length) (functions do not require direct access to memory block from the program)
OX9162_MemWrite	function for writing the card register (8-bit data length) (functions do not require direct access to memory block from the program)
Interrupt handling functions (open connection to the card required)	
function	**description**
OX9162_IRQ_Wait	stops the thread until the interrupt event occurs or timeout period expires
OX9162_IRQ_SetTimeout	sets the timeout period used by the OX9162_IRQ_Wait function
OX9162_IRQ_Count	passes the number of interrupt events triggered by the card

A1.2 Return codes of functions (tedia_ox9162.dll library version 3.00 and later)

The table below provides an overview of all return codes used by the library functions.

Return codes overview	
return code	**description**
OX9162_Ok	The function was terminated without errors.
OX9162_BadIndex	The specified index is either outside the allowed range, or there is no available card with this index number.
OX9162_AlreadyOpened	The connection to the card has already been opened by another process.
OX9162_OpenedSameProcess	The connection to the card has already been opened within of the current process.
OX9162_CardOpenErr	An unexpected error occurred while opening the connection to the card.
OX9162_IntEnableErr	An error occurred while trying to enable interrupts functionality for the card.
OX9162_NotOpened	The connection to the card has not been opened yet.
OX9162_BAR4Err	BAR4 is not available.
OX9162_IntDisabled	Interrupt functionality has not been enabled for the card (ie. the connection to the card was opened with the OX9162_Open or OX9162_OpenMulti function with the enable_int = 0 option).
OX9162_StopIntThread	The last interrupt event or timeout, it is necessary to stop the execution of the thread because the connection to the card has been closed. This return code does not indicate an error, it is for information only.
OX9162_NullPointer	A null pointer passed within the function parameters.
OX9162_CreateEvent	An unexpected error occurred while waiting for the interrupt event.
OX9162_WaitTimeout	The OX9162_IRQ_Wait function was terminated by the expiration of the preconfigured timeout period.
OX9162_VersionConflict	Conflict between the system driver and the driver interface library.

A2.1 OX9162_InterfaceVersion

DWORD OX9162_InterfaceVersion (VOID);

Description

The function provides the driver interface version number (ie. the tedia_ox9162.dll library); for example, in the case of version 3.06, the value 306 is passed, so it is necessary to shift two decimal places to the right before displaying.

The function does not require a previous opening of connection to any card.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox9162.dll library (moreover, it does not require the availability of any card in system).

Return codes

The function does not use the return code method because the returned value directly passes required information.

A2.2 OX9162_DriverVersion

OX9162_RET OX9162_DriverVersion (DWORD **ver*);

****ver***

Pointer to the memory location where the "driver version" value was stored by the function.

Description

The function provides the system driver version number (ie. the tedia_OxPCI.sys); for example, in the case of version 1.10, the value 110 is passed, so it is necessary to shift two decimal places to the right before displaying.

The function does not require a previous opening of connection to any card.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox9162.dll library (moreover, it does not require the availability of any card in system).

Return codes

In the case of successful termination, the function passes a return code *OX9162_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

A2.3 OX9162_CardsInSystem

OX9162_RET OX9162_CardsInSystem (DWORD **count*);

****count***

Pointer to the memory location where the "number of card" value was stored by the function.

Description

The function provides the number of cards available to the tedia_ox9162.dll library.

The value from 1 upwards indicates the actual number of cards available on the computer (zero value means that there is no card in the system).

The function does not require a previous opening of connection to any card.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox9162.dll library (moreover, it does not require the availability of any card in system).

Return codes

In the case of successful termination, the function passes a return code *OX9162_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

A2.4 OX9162_Card_DID

```
OX9162_RET  OX9162_Card_DID          ( DWORD  inx,
                                       DWORD  *DID );
```

inx

Defines the card selected from the list of available cards.

****DID***

Pointer to the memory location where the "PCI Device ID" value was stored by the function.

Description

The function provides the PCI Device ID number of card selected by the *inx* (in connection with TEDIA VID uniquely defines the type of card).

The program selects the card from the list of available cards (they are numbered from zero), so the parameter *inx* must be in the range from 0 to the value passed by *OX9162_CardInSystem* reduced by 1 (ie. in case of two cards in the computer *OX9162_CardInSystem* returns 2 and parameter *inx* becomes valid for values 0 and 1).

The *DID* value corresponds to the PCI Device ID (16-bit integer) assigned to the card type by the manufacturer.

The function does not require a previous opening of connection to any card.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox9162.dll library.

Return codes

In the case of successful termination, the function passes a return code *OX9162_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

A2.5 OX9162_Card_BUS

```
OX9162_RET  OX9162_Card_BUS          ( DWORD  inx,
                                       DWORD  *bus );
```

inx

Defines the card selected from the list of available cards.

****bus***

Pointer to the memory location where the "PCI bus number" value was stored by the function.

Description

The function provides the PCI bus number of card selected by the *inx*, where the card is installed. In connection with PCI slot number passed by *OX9162_Card_SLOT* function uniquely defines the location of the card in the computer (the set of bus and slot values is unique in the computer).

The program selects the card from the list of available cards (they are numbered from zero), so the parameter *inx* must be in the range from 0 to the value passed by *OX9162_CardInSystem* reduced by 1 (ie. in case of two cards in the computer *OX9162_CardInSystem* returns 2 and parameter *inx* becomes valid for values 0 and 1).

The function does not require a previous opening of connection to any card.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox9162.dll library.

Return codes

In the case of successful termination, the function passes a return code *OX9162_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

A2.6 OX9162_Card_SLOT

```
OX9162_RET  OX9162_Card_SLOT        ( DWORD  inx,
                                      DWORD  *slot );
```

inx

Defines the card selected from the list of available cards.

****slot***

Pointer to the memory location where the "PCI slot number" value was stored by the function.

Description

The function provides the PCI slot number of card selected by the *inx*, where the card is installed. In connection with PCI bus number passed by *OX9162_Card_BUS* function uniquely defines the location of the card in the computer (the set of bus and slot values is unique in the computer).

The program selects the card from the list of available cards (they are numbered from zero), so the parameter *inx* must be in the range from 0 to the value passed by *OX9162_CardInSystem* reduced by 1 (ie. in case of two cards in the computer *OX9162_CardInSystem* returns 2 and parameter *inx* becomes valid for values 0 and 1).

The function does not require a previous opening of connection to any card.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox9162.dll library.

Return codes

In the case of successful termination, the function passes a return code *OX9162_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

A2.7 OX9162_Open

```
OX9162_RET  OX9162_Open          ( DWORD  inx,
                                   BOOL  enable_int );
```

inx

Defines the card selected from the list of available cards.

enable_int

The parameter enables the card interrupt functionality with the value true (false value disables this capability).

Description

This function opens an exclusive connection to the card selected by *inx*.

The program selects the card from the list of available cards (they are numbered from zero), so the parameter *inx* must be in the range from 0 to the value passed by *OX9162_CardInSystem* reduced by 1 (ie. in case of two cards in the computer *OX9162_CardInSystem* returns 2 and parameter *inx* becomes valid for values 0 and 1).

The *enable_int* parameter enables the interrupt functionality of the card with the value true; if the card does not support this interrupt capability, the return code *OX9162_IntEnableErr* is passed when attempting to enable the interrupt. Note, that all standard cards support interrupt capability, and the false option is only a reserve for non-standard types.

Access to the card is exclusive; when trying to open a connection by this *OX9162_Open* function to a card that is currently being used by another process (ie. another process has opened connection and not closed it yet; it does not matter whether the connection was opened with *OX9162_Open* or *OX9162_OpenMulti*), the return code *OX9162_AlreadyOpened* is passed.

When trying to open a connection by this *OX9162_Open* function to a card that is currently being used by the same process, the return code *OX9162_OpenedSameProcess* is passed to indicate that the connection was not opened.

If simultaneous access to the card from multiple processes is needed, the *OX9162_OpenMulti* function can be used instead of the *OX9162_Open* function.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox9162.dll library.

Return codes

In the case of successful termination, the function passes a return code *OX9162_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

A2.8 OX9162_OpenMulti (interface version 3.03 and later)

OX9162_RET OX9162_OpenMulti (DWORD *inx*,
BOOL *enable_int*);

inx

Defines the card selected from the list of available cards.

enable_int

The parameter enables the card interrupt functionality with the value true (false value disables this capability).

Description

This function opens a non-exclusive connection to the card selected by *inx*.

The program selects the card from the list of available cards (they are numbered from zero), so the parameter *inx* must be in the range from 0 to the value passed by *OX9162_CardInSystem* reduced by 1 (ie. in case of two cards in the computer *OX9162_CardInSystem* returns 2 and parameter *inx* becomes valid for values 0 and 1).

The *enable_int* parameter enables the interrupt functionality of the card with the value true; if the card does not support this interrupt capability, the return code *OX9162_IntEnableErr* is passed when attempting to enable the interrupt. Note, that all standard PCI cards support interrupt capability, and the false option is only a reserve for non-standard types.

Access to the card is non-exclusive; when trying to open a connection by this *OX9162_OpenMulti* function to a card that is currently being used non-exclusive by another process (ie. another process has opened connection by *OX9162_OpenMulti* and not closed it yet), the return code *OX9162_OK* is passed and the connection to the card will be open. However, when trying to open a connection by this *OX9162_OpenMulti* function to a card that is currently being used exclusive by another process (ie. another process has opened connection by *OX9162_Open* and not closed it yet), the return code *OX9162_AlreadyOpened* is passed to indicate that the connection was not opened.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox9162.dll library (starting with 3.03).

Return codes

In the case of successful termination, the function passes a return code *OX9162_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

Warning: *It is necessary to realize that the control rights of all processes are symmetrical and in the case of access to the same functions/registers they can cause mutual collisions. If it is necessary to ensure simultaneous access from several programs (or processes), it is better to use the OX9162_Open function to open a exclusive connection to the card, blocking the simultaneous opening of another program.*

A2.9 OX9162_Card_OpenCount (interface version 3.03 and later)

```
OX9162_RET  OX9162_Card_OpenCount   ( DWORD  inx,
                                      DWORD  *count );
```

inx

Defines the card selected from the list of available cards.

****count***

Pointer to the memory location where the "current number of open connection" value was stored by the function.

Description

The function provides the "current number of open connection" value established by *OX9162_OpenMulti* (value from one above) or *OX9162_Open* function (value always equal to one).

The *inx* parameter must match the index of the card to which the process has already opened a connection.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox9162.dll library (starting with 3.03).

Return codes

In the case of successful termination, the function passes a return code *OX9162_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

A2.10 OX9162_Close

OX9162_RET OX9162_Close (DWORD *inx*);

inx

Defines the card selected from the list of available cards.

Description

The function closes the connection to the card specified with the index *inx*.

The *inx* parameter must match the index of the card to which the process has already opened a connection.

Note: When close a running program, connection between this program and the card is automatically closed (and also if connections with more than one card exists, they are all closed).

Supported cards

This function is implemented for all types of cards supported by the tedia_ox9162.dll library.

Return codes

In the case of successful termination, the function passes a return code *OX9162_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

A2.11 OX9162_BAR4_BaseAdr

```
OX9162_RET  OX9162_BAR4_BaseAdr     ( DWORD  inx,
                                      size_t  *base_adr,
                                      DWORD  *range );
```

inx

Defines the card selected from the list of available cards.

****base_adr***

Pointer to the memory location where the "BAR4 base address" value was stored by the function.

****range***

Pointer to the memory location where the "BAR4 registers block range" value was stored by the function.

Description

The function provides the "BAR4 base address" and "BAR4 registers block range" values, ie. starting address and block size of the memory for accessing the user function registers.

The *inx* parameter must match the index of the card to which the process has already opened a connection.

Note: To access the registers a pair of functions OX9162_MemRead and OX9162_MemWrite can be used.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox9162.dll library.

Return codes

In the case of successful termination, the function passes a return code *OX9162_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

A2.12 OX9162_MemRead

```
BYTE  OX9162_MemRead                ( size_t  base_adr,
                                      DWORD  offset );
```

base_adr

Specifies the starting address of the memory block for accessing the card registers (ie. the address detected by the *OX9162_BAR4_BaseAdr* function).

offset

Specifies the address offset of the required card register within the memory block.

Description

The function is used to read the contents of the any card's registers, especially in the case of development environments that do not support direct memory access (eg. Visual Basic).

The actual address accessed by the function is given by the sum of the *base_adr* and *offset* values.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox9162.dll library.

Return codes

The function does not use the return code method because the returned value directly passes required information.

Warning: *The function does not disable access to addresses outside the allocated space of the card and thus causing a fatal program runtime error; therefore, all responsibility for the correct address lies with the program.*

A2.13 OX9162_MemWrite

```
VOID  OX9162_MemWrite          ( size_t  base_adr,
                                 DWORD  offset,
                                 BYTE  data );
```

base_adr

Specifies the starting address of the memory block for accessing the card registers (ie. the address detected by the *OX9162_BAR4_BaseAdr* function).

offset

Specifies the address offset of the required card register within the memory block.

data

Defines the value written to the card register.

Description

The function is used to write the contents to the any card's registers, especially in the case of development environments that do not support direct memory access (eg. Visual Basic).

The actual address accessed by the function is given by the sum of the *base_adr* and *offset* values.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox9162.dll library.

Return codes

The function does not use the return code method.

Warning: *The function does not disable access to addresses outside the allocated space of the card and thus causing a fatal program runtime error; therefore, all responsibility for the correct address lies with the program.*

A2.14 OX9162_IRQ_Wait

```
OX9162_RET  OX9162_IRQ_Wait          ( DWORD  inx,
                                       DWORD  * );
```

inx

Defines the card selected from the list of available cards.

*

The second parameter has not been used since version 1.55 (maintained for backward compatibility).

Description

The function suspends the thread until the driver detects the interrupt event or until the interrupt timeout period expires (see the description of the *OX9162_IRQ_SetTimeout* function).

The *inx* parameter must match the index of the card to which the process has already opened a connection.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox9162.dll library.

Return codes

In the case of successful termination, the function passes a return code *OX9162_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

The return code *OX9162_WaitTimeout* indicates that the function was terminated without errors, but no event caused an interruption of system and function was terminated by the expiration of the preconfigured timeout period.

However, if the function passes the return code *OX9162_StopInThread*, it is necessary to stop the execution of the thread because the connection to the card has been probably closed.

A2.15 OX9162_IRQ_SetTimeout

```
OX9162_RET  OX9162_IRQ_SetTimeout    ( DWORD  inx,
                                       DWORD  timeout );
```

inx

Defines the card selected from the list of available cards.

timeout

Defines the interrupt timeout period (ie. latency time) for which the *OX9162_IRQ_Wait* function suspends a thread. The value is entered in milliseconds.

Description

This function sets the timeout period (ie. waiting limit for an interrupt event) used by the *OX9162_IRQ_Wait* function. If the interrupt event is not detect before the timeout period expires, the *OX9162_IRQ_Wait* function is terminated (ie the suspended thread is released) with the return code *OX9162_WaitTimeout*.

If the *timeout* value is set to zero, the *OX9162_IRQ_Wait* function detects whether an interrupt event has been detected since the *OX9162_IRQ_Wait* function was previously executed, and without further waiting, the function is terminated with the return code *OX9162_OK* or *OX9162_WaitTimeout*.

The *inx* parameter must match the index of the card to which the process has already opened a connection.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox9162.dll library.

Return codes

In the case of successful termination, the function passes a return code *OX9162_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

A2.16 OX9162_IRQ_Count

DWORD OX9162_IRQ_Count (DWORD *inx*);

inx

Defines the card selected from the list of available cards.

Description

The function provides the "the number of received interrupts" value for a card specified by the index *inx*.

The *OX9162_IRQ_Count* function can also be used if interrupt capability was not enabled when the *OX9162_OpenMulti* function connection was opened; the reason is because there may be situations where another program opens (or has already opened) a connection to the card with an enabled interrupt capability.

The *inx* parameter must match the index of the card to which the process has already opened a connection.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox9162.dll library.

Return codes

The function does not use the return code method because the returned value directly passes required information.

B1.1 Function set of the tedia_ox952.dll library version 3.00 and later

The table below provides an overview of all public functions implemented in library, a detailed description follows on next pages of this appendix.

General functions independent of installed cards	
function	**description**
OX952_InterfaceVersion	driver interface version (ie. tedia_ox952.dll)
OX952_DriverVersion	system driver version (ie. tedia_OxPCI.sys)
OX952_CardsInSystem	number of cards available to the tedia_ox952.dll library
Function to determine the properties of cards (usable even before opening the connection to the card)	
function	**description**
OX952_Card_DID	the PCI Device ID of the selected card (in conjunction with TEDIA VID defines the type of card)
OX952_Card_BUS	the PCI bus number where the selected card is installed
OX952_Card_SLOT	the PCI slot number where the selected card is installed
Functions for opening/closing the connection to the card	
function	**description**
OX952_Open	opens an exclusive connection to the selected card (ie. another program cannot open the connection using OX952_OpenMulti, nor OX952_Open)
OX952_OpenMulti	opens an non-exclusive connection to the selected card (ie. another program can open the connection using OX952_OpenMulti, but cannot open the connection using OX952_Open)
OX952_Card_OpenCount	current number of open connection using OX952_OpenMulti function
OX952_Close	closes the connection to the card
Functions to access to the card registers (open connection to the card required)	
function	**description**
OX952_BAR05_BaseAdr	the starting address and size of the memory block mapping the functional registers of the card (BAR5, PCI function 0); necessary for direct register access (this feature is not required for common applications)
OX952_BAR11_BaseAdr	the starting address and size of the memory block mapping the functional registers of the card (BAR1, PCI function 1); necessary for direct register access
OX952_MemRead	function for reading the card register (8-bit data length) (functions do not require direct access to memory block from the program)
OX952_MemWrite	function for writing the card register (8-bit data length) (functions do not require direct access to memory block from the program)
Interrupt handling functions (open connection to the card required)	
function	**description**
OX952_IRQ_Wait	stops the thread until the interrupt event occurs or timeout period expires
OX952_IRQ_SetTimeout	sets the timeout period used by the OX952_IRQ_Wait function
OX952_IRQ_Count	passes the number of interrupt events triggered by the card

B1.2 Return codes of functions (tedia_ox952.dll library version 3.00 and later)

The table below provides an overview of all return codes used by the library functions.

Return codes overview	
return code	**description**
OX952_Ok	The function was terminated without errors.
OX952_BadIndex	The specified index is either outside the allowed range, or there is no available card with this index number.
OX952_AlreadyOpened	The connection to the card has already been opened by another process.
OX952_OpenedSameProcess	The connection to the card has already been opened within of the current process.
OX952_CardOpenErr	An unexpected error occurred while opening the connection to the card.
OX952_IntEnableErr	An error occurred while trying to enable interrupts functionality for the card.
OX952_NotOpened	The connection to the card has not been opened yet.
OX952_BAR05Err	BAR5 (PCI function 0) is not available.
OX952_BAR11Err	BAR1 (PCI function 1) is not available.
OX952_IntDisabled	Interrupt functionality has not been enabled for the card (ie. the connection to the card was opened with the OX952_Open or OX952_OpenMulti function with the enable_int = 0 option).
OX952_StopIntThread	The last interrupt event or timeout, it is necessary to stop the execution of the thread because the connection to the card has been closed. This return code does not indicate an error, it is for information only.
OX952_NullPointer	A null pointer passed within the function parameters.
OX952_CreateEvent	An unexpected error occurred while waiting for the interrupt event.
OX952_WaitTimeout	The OX952_IRQ_Wait function was terminated by the expiration of the preconfigured timeout period.
OX952_VersionConflict	Conflict between the system driver and the driver interface library.
OX952_NoCoupledDevice	Error while pairing of PCI functions within one card (probably the system driver was not installed correctly for both PCI functions).

B2.1 OX952_InterfaceVersion

DWORD OX952_InterfaceVersion (VOID);

Description

The function provides the driver interface version number (ie. the tedia_ox952.dll library); for example, in the case of version 3.06, the value 306 is passed, so it is necessary to shift two decimal places to the right before displaying.

The function does not require a previous opening of connection to any card.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox952.dll library (moreover, it does not require the availability of any card in system).

Return codes

The function does not use the return code method because the returned value directly passes required information.

B2.2 OX952_DriverVersion

OX952_RET OX952_DriverVersion (DWORD **ver*);

****ver***

Pointer to the memory location where the "driver version" value was stored by the function.

Description

The function provides the system driver version number (ie. the tedia_OxPCI.sys); for example, in the case of version 1.10, the value 110 is passed, so it is necessary to shift two decimal places to the right before displaying.

The function does not require a previous opening of connection to any card.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox952.dll library (moreover, it does not require the availability of any card in system).

Return codes

In the case of successful termination, the function passes a return code *OX952_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

B2.3 OX952_CardsInSystem

OX952_RET OX952_CardsInSystem (DWORD **count*);

****count***

Pointer to the memory location where the "number of card" value was stored by the function.

Description

The function provides the number of cards available to the tedia_ox952.dll library.

The value from 1 upwards indicates the actual number of cards available on the computer (zero value means that there is no card in the system).

The function does not require a previous opening of connection to any card.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox952.dll library (moreover, it does not require the availability of any card in system).

Return codes

In the case of successful termination, the function passes a return code *OX952_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

B2.4 OX952_Card_DID

```
OX952_RET  OX952_Card_DID          ( DWORD  inx,
                                     DWORD  *DID );
```

inx

Defines the card selected from the list of available cards.

****DID***

Pointer to the memory location where the "PCI Device ID" value was stored by the function.

Description

The function provides the PCI Device ID number of card selected by the *inx* (in connection with TEDIA VID uniquely defines the type of card).

The program selects the card from the list of available cards (they are numbered from zero), so the parameter *inx* must be in the range from 0 to the value passed by *OX952_CardInSystem* reduced by 1 (ie. in case of two cards in the computer *OX952_CardInSystem* returns 2 and parameter *inx* becomes valid for values 0 and 1).

The *DID* value corresponds to the PCI Device ID (16-bit integer) assigned to the card type by the manufacturer.

The function does not require a previous opening of connection to any card.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox952.dll library.

Return codes

In the case of successful termination, the function passes a return code *OX952_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

B2.5 OX952_Card_BUS

```
OX952_RET  OX952_Card_BUS          ( DWORD  inx,
                                     DWORD  *bus );
```

inx

Defines the card selected from the list of available cards.

****bus***

Pointer to the memory location where the "PCI bus number" value was stored by the function.

Description

The function provides the PCI bus number of card selected by the *inx*, where the card is installed. In connection with PCI slot number passed by *OX952_Card_SLOT* function uniquely defines the location of the card in the computer (the set of bus and slot values is unique in the computer).

The program selects the card from the list of available cards (they are numbered from zero), so the parameter *inx* must be in the range from 0 to the value passed by *OX952_CardInSystem* reduced by 1 (ie. in case of two cards in the computer *OX952_CardInSystem* returns 2 and parameter *inx* becomes valid for values 0 and 1).

The function does not require a previous opening of connection to any card.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox952.dll library.

Return codes

In the case of successful termination, the function passes a return code *OX952_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

B2.6 OX952_Card_SLOT

```
OX952_RET  OX952_Card_SLOT          ( DWORD  inx,
                                      DWORD  *slot );
```

inx

Defines the card selected from the list of available cards.

****slot***

Pointer to the memory location where the "PCI slot number" value was stored by the function.

Description

The function provides the PCI slot number of card selected by the *inx*, where the card is installed. In connection with PCI bus number passed by *OX952_Card_BUS* function uniquely defines the location of the card in the computer (the set of bus and slot values is unique in the computer).

The program selects the card from the list of available cards (they are numbered from zero), so the parameter *inx* must be in the range from 0 to the value passed by *OX952_CardInSystem* reduced by 1 (ie. in case of two cards in the computer *OX952_CardInSystem* returns 2 and parameter *inx* becomes valid for values 0 and 1).

The function does not require a previous opening of connection to any card.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox952.dll library.

Return codes

In the case of successful termination, the function passes a return code *OX952_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

B2.7 OX952_Open

```
OX952_RET  OX952_Open          ( DWORD  inx,
                                 BOOL  enable_int );
```

inx

Defines the card selected from the list of available cards.

enable_int

The parameter enables the card interrupt functionality with the value true (false value disables this capability).

Description

This function opens an exclusive connection to the card selected by *inx*.

The program selects the card from the list of available cards (they are numbered from zero), so the parameter *inx* must be in the range from 0 to the value passed by *OX952_CardInSystem* reduced by 1 (ie. in case of two cards in the computer *OX952_CardInSystem* returns 2 and parameter *inx* becomes valid for values 0 and 1).

The *enable_int* parameter enables the interrupt functionality of the card with the value true; if the card does not support this interrupt capability, the return code *OX952_IntEnableErr* is passed when attempting to enable the interrupt. Note, that all standard cards support interrupt capability, and the false option is only a reserve for non-standard types.

Access to the card is exclusive; when trying to open a connection by this *OX952_Open* function to a card that is currently being used by another process (ie. another process has opened connection and not closed it yet; it does not matter whether the connection was opened with *OX952_Open* or *OX952_OpenMulti*), the return code *OX952_AlreadyOpened* is passed.

When trying to open a connection by this *OX952_Open* function to a card that is currently being used by the same process, the return code *OX952_OpenedSameProcess* is passed to indicate that the connection was not opened.

If simultaneous access to the card from multiple processes is needed, the *OX952_OpenMulti* function can be used instead of the *OX952_Open* function.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox952.dll library.

Return codes

In the case of successful termination, the function passes a return code *OX952_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

B2.8 OX952_OpenMulti (interface version 3.03 and later)

```
OX952_RET  OX952_OpenMulti          ( DWORD  inx,
                                      BOOL  enable_int );
```

inx

Defines the card selected from the list of available cards.

enable_int

The parameter enables the card interrupt functionality with the value true (false value disables this capability).

Description

This function opens a non-exclusive connection to the card selected by *inx*.

The program selects the card from the list of available cards (they are numbered from zero), so the parameter *inx* must be in the range from 0 to the value passed by *OX952_CardInSystem* reduced by 1 (ie. in case of two cards in the computer *OX952_CardInSystem* returns 2 and parameter *inx* becomes valid for values 0 and 1).

The *enable_int* parameter enables the interrupt functionality of the card with the value true; if the card does not support this interrupt capability, the return code *OX952_IntEnableErr* is passed when attempting to enable the interrupt. Note, that all standard PCI cards support interrupt capability, and the false option is only a reserve for non-standard types.

Access to the card is non-exclusive; when trying to open a connection by this *OX952_OpenMulti* function to a card that is currently being used non-exclusive by another process (ie. another process has opened connection by *OX952_OpenMulti* and not closed it yet), the return code *OX952_OK* is passed and the connection to the card will be open. However, when trying to open a connection by this *OX952_OpenMulti* function to a card that is currently being used exclusive by another process (ie. another process has opened connection by *OX952_Open* and not closed it yet), the return code *OX952_AlreadyOpened* is passed to indicate that the connection was not opened.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox952.dll library (starting with 3.03).

Return codes

In the case of successful termination, the function passes a return code *OX952_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

Warning: *It is necessary to realize that the control rights of all processes are symmetrical and in the case of access to the same functions/registers they can cause mutual collisions. If it is necessary to ensure simultaneous access from several programs (or processes), it is better to use the OX952_Open function to open a exclusive connection to the card, blocking the simultaneous opening of another program.*

B2.9 OX952_Card_OpenCount (interface version 3.03 and later)

```
OX952_RET  OX952_Card_OpenCount      ( DWORD  inx,
                                       DWORD  *count );
```

inx

Defines the card selected from the list of available cards.

****count***

Pointer to the memory location where the "current number of open connection" value was stored by the function.

Description

The function provides the "current number of open connection" value established by *OX952_OpenMulti* (value from one above) or *OX952_Open* function (value always equal to one).

The *inx* parameter must match the index of the card to which the process has already opened a connection.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox952.dll library (starting with 3.03).

Return codes

In the case of successful termination, the function passes a return code *OX952_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

7B2.10 OX952_Close

OX952_RET OX952_Close (DWORD *inx*);

inx

Defines the card selected from the list of available cards.

Description

The function closes the connection to the card specified with the index *inx*.

The *inx* parameter must match the index of the card to which the process has already opened a connection.

Note: When close a running program, connection between this program and the card is automatically closed (and also if connections with more than one card exists, they are all closed).

Supported cards

This function is implemented for all types of cards supported by the tedia_ox952.dll library.

Return codes

In the case of successful termination, the function passes a return code *OX952_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

B2.11 OX952_BAR05_BaseAdr, OX952_BAR11_BaseAdr

```
OX952_RET  OX952_BAR05_BaseAdr    ( DWORD  inx,
                                    size_t  *base_adr,
                                    DWORD  *range );
OX952_RET  OX952_BAR11_BaseAdr    ( DWORD  inx,
                                    size_t  *base_adr,
                                    DWORD  *range );
```

inx

Defines the card selected from the list of available cards.

****base_adr***

Pointer to the memory location where the "BARxx base address" value was stored by the function.

****range***

Pointer to the memory location where the "BARxx registers block range" value was stored by the function.

Description

The function *OX952_BAR05_BaseAdr* provides the "BAR05 base address" and "BAR05 registers block range" values, ie. starting address and block size of the memory for accessing the serial port registers.

Note: The serial port is dedicated primarily for service purposes (ie. user-friendly updating the card firmware), but it also allows to determine the unique serial number of the card.

The function *OX952_BAR11_BaseAdr* provides the "BAR11 base address" and "BAR11 registers block range" values, ie. starting address and block size of the memory for accessing the user function registers.

The *inx* parameter must match the index of the card to which the process has already opened a connection.

Note: To access the registers a pair of functions OX952_MemRead and OX952_MemWrite can be used.

Supported cards

The function *OX952_BAR05_BaseAdr* is implemented for most types of cards supported by the tedia_ox952.dll library (see appendix D of this manual).

The function *OX952_BAR11_BaseAdr* is implemented for all types of cards supported by the tedia_ox952.dll library.

Return codes

In the case of successful termination, the function passes a return code *OX952_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

B2.12 OX952_MemRead

```
BYTE  OX952_MemRead                ( size_t  base_adr,
                                     DWORD  offset );
```

base_adr

Specifies the starting address of the memory block for accessing the card registers (ie. the address detected by the *OX952_BAR05_BaseAdr* or *OX952_BAR11_BaseAdr* functions).

offset

Specifies the address offset of the required card register within the memory block (this value is described in the card's programming manual).

Description

The function is used to read the contents of the any card's registers, especially in the case of development environments that do not support direct memory access (eg. Visual Basic).

The actual address accessed by the function is given by the sum of the *base_adr* and *offset* values.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox952.dll library.

Return codes

The function does not use the return code method because the returned value directly passes required information.

Warning: *The function does not disable access to addresses outside the allocated space of the card and thus causing a fatal program runtime error; therefore, all responsibility for the correct address lies with the program.*

B2.13 OX952_MemWrite

```
VOID  OX952_MemWrite          ( size_t  base_adr,
                                DWORD  offset,
                                BYTE  data );
```

base_adr

Specifies the starting address of the memory block for accessing the card registers (ie. the address detected by the *OX952_BAR05_BaseAdr* or *OX952_BAR11_BaseAdr* functions).

offset

Specifies the address offset of the required card register within the memory block (this value is described in the card's programming manual).

data

Defines the value written to the card register.

Description

The function is used to write the contents to the any card's registers, especially in the case of development environments that do not support direct memory access (eg. Visual Basic).

The actual address accessed by the function is given by the sum of the *base_adr* and *offset* values.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox952.dll library.

Return codes

The function does not use the return code method.

Warning: *The function does not disable access to addresses outside the allocated space of the card and thus causing a fatal program runtime error; therefore, all responsibility for the correct address lies with the program.*

B2.14 OX952_IRQ_Wait

```
OX952_RET  OX952_IRQ_Wait          ( DWORD  inx,
                                     DWORD  * );
```

inx

Defines the card selected from the list of available cards.

*

The second parameter has not been used since version 1.55 (maintained for backward compatibility).

Description

The function suspends the thread until the driver detects the interrupt event or until the interrupt timeout period expires (see the description of the *OX952_IRQ_SetTimeout* function).

The *inx* parameter must match the index of the card to which the process has already opened a connection.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox952.dll library.

Return codes

In the case of successful termination, the function passes a return code *OX952_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

The return code *OX952_WaitTimeout* indicates that the function was terminated without errors, but no event caused an interruption of system and function was terminated by the expiration of the preconfigured timeout period.

However, if the function passes the return code *OX952_StopInThread*, it is necessary to stop the execution of the thread because the connection to the card has been probably closed.

B2.15 OX952_IRQ_SetTimeout

```
OX952_RET  OX952_IRQ_SetTimeout      ( DWORD  inx,
                                       DWORD  timeout );
```

inx

Defines the card selected from the list of available cards.

timeout

Defines the interrupt timeout period (ie. latency time) for which the *OX952_IRQ_Wait* function suspends a thread. The value is entered in milliseconds.

Description

This function sets the timeout period (ie. waiting limit for an interrupt event) used by the *OX952_IRQ_Wait* function. If the interrupt event is not detect before the timeout period expires, the *OX952_IRQ_Wait* function is terminated (ie the suspended thread is released) with the return code *OX952_WaitTimeout*.

If the *timeout* value is set to zero, the *OX952_IRQ_Wait* function detects whether an interrupt event has been detected since the *OX952_IRQ_Wait* function was previously executed, and without further waiting, the function is terminated with the return code *OX952_OK* or *OX952_WaitTimeout*.

The *inx* parameter must match the index of the card to which the process has already opened a connection.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox952.dll library.

Return codes

In the case of successful termination, the function passes a return code *OX952_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

B2.16 OX952_IRQ_Count

DWORD OX952_IRQ_Count (DWORD *inx*);

inx

Defines the card selected from the list of available cards.

Description

The function provides the "the number of received interrupts" value for a card specified by the index *inx*.

The *OX952_IRQ_Count* function can also be used if interrupt capability was not enabled when the *OX952_OpenMulti* function connection was opened; the reason is because there may be situations where another program opens (or has already opened) a connection to the card with an enabled interrupt capability.

The *inx* parameter must match the index of the card to which the process has already opened a connection.

Supported cards

This function is implemented for all types of cards supported by the tedia_ox952.dll library.

Return codes

The function does not use the return code method because the returned value directly passes required information.

C1.1 Function set of the tedia_ep4gxa.dll library version 3.10 and later

The table below provides an overview of all public functions implemented in library, a detailed description follows on next pages of this appendix.

General functions independent of installed cards	
function	**description**
EP4GXA_InterfaceVersion	driver interface version (ie. tedia_ep4gxa.dll)
EP4GXA_DriverVersion	system driver version (ie. tedia_OxPCI.sys)
EP4GXA_CardsInSystem	number of cards available to the tedia_ep4gxa.dll library
Function to determine the properties of cards (usable even before opening the connection to the card)	
function	**description**
EP4GXA_Card_DID	the PCI Device ID of the selected card (in conjunction with TEDIA VID defines the type of card)
EP4GXA_Card_BUS	the PCI bus number where the selected card is installed
EP4GXA_Card_SLOT	the PCI slot number where the selected card is installed
EP4GXA_Card_ID (version 3.18 and later)	the CardID value (defined by on-board DIP switch position)
EP4GXA_Card_SerNr (version 3.18 and later)	the serial number value
Functions for opening/closing the connection to the card	
function	**description**
EP4GXA_Open	opens an exclusive connection to the selected card (ie. another program cannot open the connection using EP4GXA_Open, nor EP4GXA_OpenMulti)
EP4GXA_OpenMulti	opens an non-exclusive connection to the selected card (ie. another program can open the connection using EP4GXA_OpenMulti, but cannot open the connection using EP4GXA_Open)
EP4GXA_Card_OpenCount	current number of open connection using EP4GXA_OpenMulti function
EP4GXA_Close	closes the connection to the card
Functions to access to the card registers (open connection to the card required)	
function	**description**
EP4GXA_BAR0_BaseAdr	the starting address and size of the memory block mapping the functional registers of the card (BAR0); necessary for direct register access
EP4GXA_ReadReg8 EP4GXA_ReadReg32	two functions for reading the card register in variants for 8-bit or 32-bit data (functions do not require direct access to memory block from the program)
EP4GXA_WriteReg8 EP4GXA_WriteReg32	two functions for writing to the card register in variants for 8-bit or 32-bit data (functions do not require direct access to memory block from the program)
EP4GXA_ReadReg64 EP4GXA_ReadReg2x32 (version 3.18 and later)	two functions for simultaneous reading two card registers in variants for 64-bit data or a pair of 32-bit data (functions do not require direct access to memory block from the program)
EP4GXA_WriteReg64 EP4GXA_WriteReg2x32 (version 3.18 and later)	two functions for simultaneous writing to two card registers in variants for 64-bit data or a pair of 32-bit data (functions do not require direct access to memory block from the program)
Interrupt handling functions (open connection to the card required)	
function	**description**
EP4GXA_IRQ_Wait	stops the thread until the interrupt event occurs or timeout period expires
EP4GXA_IRQ_SetTimeout	sets the timeout period used by the EP4GXA_IRQ_Wait function
EP4GXA_IRQ_Count	passes the number of interrupt events triggered by the card

C1.2 Return codes of functions (tedia_ep4gxa.dll library version 3.10 and later)

The table below provides an overview of all return codes used by the library functions.

Return codes overview	
return code	**description**
EP4GXA_Ok	The function was terminated without errors.
EP4GXA_BadIndex	The specified index is either outside the allowed range, or there is no available card with this index number.
EP4GXA_AlreadyOpened	The connection to the card has already been opened by another process.
EP4GXA_OpenedSameProcess	The connection to the card has already been opened within of the current process.
EP4GXA_CardOpenErr	An unexpected error occurred while opening the connection to the card.
EP4GXA_IntEnableErr	An error occurred while trying to enable interrupts functionality for the card.
EP4GXA_NotOpened	The connection to the card has not been opened yet.
EP4GXA_BARErr	BAR0 is not available (or other BARs for service purposes are not available).
EP4GXA_IntDisabled	Interrupt functionality has not been enabled for the card (ie. the connection to the card was opened with the EP4GXA_Open or EP4GXA_OpenMulti function with the enable_int = 0 option).
EP4GXA_StopIntThread	The last interrupt event or timeout, it is necessary to stop the execution of the thread because the connection to the card has been closed. This return code does not indicate an error, it is for information only.
EP4GXA_NullPointer	A null pointer passed within the function parameters.
EP4GXA_CreateEvent	An unexpected error occurred while waiting for the interrupt event.
EP4GXA_WaitTimeout	The EP4GXA_IRQ_Wait function was terminated by the expiration of the preconfigured timeout period.
EP4GXA_VersionConflict	Conflict between the system driver and the driver interface library.
EP4GXA_InvalidAddress	Invalid address passed as a parameter of functions EP4GXA_ReadReg8, EP4GXA_ReadReg32, EP4GXA_WriteReg8 or EP4GXA_WriteReg32

C2.1 EP4GXA_InterfaceVersion

DWORD EP4GXA_InterfaceVersion (VOID);

Description

The function provides the driver interface version number (ie. the tedia_ep4gxa.dll library); for example, in the case of version 3.06, the value 306 is passed, so it is necessary to shift two decimal places to the right before displaying.

The function does not require a previous opening of connection to any card.

Supported cards

This function is implemented for all types of cards supported by the tedia_ep4gxa.dll library (moreover, it does not require the availability of any card in system).

Return codes

The function does not use the return code method because the returned value directly passes required information.

C2.2 EP4GXA_DriverVersion

EP4GXA_RET EP4GXA_DriverVersion (DWORD **ver*);

****ver***

Pointer to the memory location where the "driver version" value was stored by the function.

Description

The function provides the system driver version number (ie. the tedia_OxPCI.sys); for example, in the case of version 1.10, the value 110 is passed, so it is necessary to shift two decimal places to the right before displaying.

The function does not require a previous opening of connection to any card.

Supported cards

This function is implemented for all types of cards supported by the tedia_ep4gxa.dll library (moreover, it does not require the availability of any card in system).

Return codes

In the case of successful termination, the function passes a return code *EP4GXA_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

C2.3 EP4GXA_CardsInSystem

EP4GXA_RET EP4GXA_CardsInSystem (DWORD **count*);

****count***

Pointer to the memory location where the "number of card" value was stored by the function.

Description

The function provides the number of cards available to the tedia_ep4gxa.dll library.

The value from 1 upwards indicates the actual number of cards available on the computer (zero value means that there is no card in the system).

The function does not require a previous opening of connection to any card.

Supported cards

This function is implemented for all types of cards supported by the tedia_ep4gxa.dll library (moreover, it does not require the availability of any card in system).

Return codes

In the case of successful termination, the function passes a return code *EP4GXA_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

C2.4 EP4GXA_Card_DID

```
EP4GXA_RET  EP4GXA_Card_DID          ( DWORD  inx,
                                       DWORD  *DID );
```

inx

Defines the card selected from the list of available cards.

****DID***

Pointer to the memory location where the "PCI Device ID" value was stored by the function.

Description

The function provides the PCI Device ID number of card selected by the *inx* (in connection with TEDIA VID uniquely defines the type of card).

The program selects the card from the list of available cards (they are numbered from zero), so the parameter *inx* must be in the range from 0 to the value passed by *EP4GXA_CardInSystem* reduced by 1 (ie. in case of two cards in the computer *EP4GXA_CardInSystem* returns 2 and parameter *inx* becomes valid for values 0 and 1).

The *DID* value corresponds to the PCI Device ID (16-bit integer) assigned to the card type by the manufacturer.

The function does not require a previous opening of connection to any card.

Supported cards

This function is implemented for all types of cards supported by the tedia_ep4gxa.dll library.

Return codes

In the case of successful termination, the function passes a return code *EP4GXA_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

C2.5 EP4GXA_Card_BUS

```
EP4GXA_RET  EP4GXA_Card_BUS          ( DWORD  inx,
                                       DWORD  *bus );
```

inx

Defines the card selected from the list of available cards.

****bus***

Pointer to the memory location where the "PCI Express bus number" value was stored by the function.

Description

The function provides the PCI Express bus number of card selected by the *inx*, where the card is installed. In connection with PCI Express slot number passed by *EP4GXA_Card_SLOT* function uniquely defines the location of the card in the computer (the set of bus and slot values is unique in the computer).

The program selects the card from the list of available cards (they are numbered from zero), so the parameter *inx* must be in the range from 0 to the value passed by *EP4GXA_CardInSystem* reduced by 1 (ie. in case of two cards in the computer *EP4GXA_CardInSystem* returns 2 and parameter *inx* becomes valid for values 0 and 1).

The function does not require a previous opening of connection to any card.

Supported cards

This function is implemented for all types of cards supported by the tedia_ep4gxa.dll library.

Return codes

In the case of successful termination, the function passes a return code *EP4GXA_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

C2.6 EP4GXA_Card_SLOT

```
EP4GXA_RET  EP4GXA_Card_SLOT        ( DWORD  inx,
                                      DWORD  *slot );
```

inx

Defines the card selected from the list of available cards.

****slot***

Pointer to the memory location where the "PCI Express slot number" value was stored by the function.

Description

The function provides the PCI Express slot number of card selected by the *inx*, where the card is installed. In connection with PCI Express bus number passed by *EP4GXA_Card_BUS* function uniquely defines the location of the card in the computer (the set of bus and slot values is unique in the computer).

The program selects the card from the list of available cards (they are numbered from zero), so the parameter *inx* must be in the range from 0 to the value passed by *EP4GXA_CardInSystem* reduced by 1 (ie. in case of two cards in the computer *EP4GXA_CardInSystem* returns 2 and parameter *inx* becomes valid for values 0 and 1).

The function does not require a previous opening of connection to any card.

Supported cards

This function is implemented for all types of cards supported by the tedia_ep4gxa.dll library.

Return codes

In the case of successful termination, the function passes a return code *EP4GXA_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

C2.7 EP4GXA_Card_ID (interface version 3.18 and later)

```
EP4GXA_RET  EP4GXA_Card_ID          ( DWORD  inx,
                                      DWORD  *id );
```

inx

Defines the card selected from the list of available cards.

****id***

Pointer to the memory location where the "Card ID" value was stored by the function.

Description

The function provides the "Card ID" of card (ie. value in the range 0 to 3 derived from on-board DIP switch position defined by the user) selected by the *inx* help with identification of card.

The function provides the PCI Express slot number of card selected by the *inx*, where the card is installed. In connection with PCI Express bus number passed by *EP4GXA_Card_BUS* function uniquely defines the location of the card in the computer (the set of bus and slot values is unique in the computer).

The program selects the card from the list of available cards (they are numbered from zero), so the parameter *inx* must be in the range from 0 to the value passed by *EP4GXA_CardInSystem* reduced by 1 (ie. in case of two cards in the computer *EP4GXA_CardInSystem* returns 2 and parameter *inx* becomes valid for values 0 and 1).

The function does not require a previous opening of connection to any card.

Supported cards

This function is implemented for all types of cards supported by the tedia_ep4gxa.dll library.

Return codes

In the case of successful termination, the function passes a return code *EP4GXA_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

C2.8 EP4GXA_Card_SerNr (interface version 3.18 and later)

```
EP4GXA_RET  EP4GXA_Card_SerNr        ( DWORD  inx,
                                       DWORD  *sernr );
```

inx

Defines the card selected from the list of available cards.

****sernr***

Pointer to the memory location where the "serial number" value was stored by the function.

Description

The function provides the serial number of card selected by the *inx* and uniquely defines the type and specific piece of card in the computer.

The program selects the card from the list of available cards (they are numbered from zero), so the parameter *inx* must be in the range from 0 to the value passed by *EP4GXA_CardInSystem* reduced by 1 (ie. in case of two cards in the computer *EP4GXA_CardInSystem* returns 2 and parameter *inx* becomes valid for values 0 and 1).

The function does not require a previous opening of connection to any card.

Supported cards

This function is implemented for all types of cards supported by the tedia_ep4gxa.dll library.

Return codes

In the case of successful termination, the function passes a return code *EP4GXA_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

C2.9 EP4GXA_Open

```
EP4GXA_RET  EP4GXA_Open          ( DWORD  inx,
                                   BOOL  enable_int );
```

inx

Defines the card selected from the list of available cards.

enable_int

The parameter enables the card interrupt functionality with the value true (false value disables this capability).

Description

This function opens an exclusive connection to the card selected by *inx*.

The program selects the card from the list of available cards (they are numbered from zero), so the parameter *inx* must be in the range from 0 to the value passed by *EP4GXA_CardInSystem* reduced by 1 (ie. in case of two cards in the computer *EP4GXA_CardInSystem* returns 2 and parameter *inx* becomes valid for values 0 and 1).

The *enable_int* parameter enables the interrupt functionality of the card with the value true; if the card does not support this interrupt capability, the return code *EP4GXA_IntEnableErr* is passed when attempting to enable the interrupt. Note, that all standard cards support interrupt capability, and the false option is only a reserve for non-standard types.

Access to the card is exclusive; when trying to open a connection by this *EP4GXA_Open* function to a card that is currently being used by another process (ie. another process has opened connection and not closed it yet; it does not matter whether the connection was opened with *EP4GXA_Open* or *EP4GXA_OpenMulti*), the return code *EP4GXA_AlreadyOpened* is passed.

When trying to open a connection by this *EP4GXA_Open* function to a card that is currently being used by the same process, the return code *EP4GXA_OpenedSameProcess* is passed to indicate that the connection was not opened.

If simultaneous access to the card from multiple processes is needed, the *EP4GXA_OpenMulti* function can be used instead of the *EP4GXA_Open* function.

Supported cards

This function is implemented for all types of cards supported by the tedia_ep4gxa.dll library.

Return codes

In the case of successful termination, the function passes a return code *EP4GXA_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

C2.10 EP4GXA_OpenMulti

EP4GXA_RET EP4GXA_OpenMulti (DWORD *inx*,
BOOL *enable_int*);

inx

Defines the card selected from the list of available cards.

enable_int

The parameter enables the card interrupt functionality with the value true (false value disables this capability).

Description

This function opens a non-exclusive connection to the card selected by *inx*.

The program selects the card from the list of available cards (they are numbered from zero), so the parameter *inx* must be in the range from 0 to the value passed by *EP4GXA_CardInSystem* reduced by 1 (ie. in case of two cards in the computer *EP4GXA_CardInSystem* returns 2 and parameter *inx* becomes valid for values 0 and 1).

The *enable_int* parameter enables the interrupt functionality of the card with the value true; if the card does not support this interrupt capability, the return code *EP4GXA_IntEnableErr* is passed when attempting to enable the interrupt. Note, that all standard PCI and PCI Express cards support interrupt capability, and the false option is only a reserve for non-standard types.

Access to the card is non-exclusive; when trying to open a connection by this *EP4GXA_OpenMulti* function to a card that is currently being used non-exclusive by another process (ie. another process has opened connection by *EP4GXA_OpenMulti* and not closed it yet), the return code *EP4GXA_OK* is passed and the connection to the card will be open. However, when trying to open a connection by this *EP4GXA_OpenMulti* function to a card that is currently being used exclusive by another process (ie. another process has opened connection by *EP4GXA_Open* and not closed it yet), the return code *EP4GXA_AlreadyOpened* is passed to indicate that the connection was not opened.

Supported cards

This function is implemented for all types of cards supported by the tedia_ep4gxa.dll library.

Return codes

In the case of successful termination, the function passes a return code *EP4GXA_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

Warning: *It is necessary to realize that the control rights of all processes are symmetrical and in the case of access to the same functions/registers they can cause mutual collisions. If it is necessary to ensure simultaneous access from several programs (or processes), it is better to use the EP4GXA_Open function to open a exclusive connection to the card, blocking the simultaneous opening of another program.*

C2.11 EP4GXA_Card_OpenCount

```
EP4GXA_RET  EP4GXA_Card_OpenCount ( DWORD  inx,
                                    DWORD  *count );
```

inx

Defines the card selected from the list of available cards.

****count***

Pointer to the memory location where the "current number of open connection" value was stored by the function.

Description

The function provides the "current number of open connection" value established by *EP4GXA_OpenMulti* (value from one above) or *EP4GXA_Open* function (value always equal to one).

The *inx* parameter must match the index of the card to which the process has already opened a connection.

Supported cards

This function is implemented for all types of cards supported by the tedia_ep4gxa.dll library.

Return codes

In the case of successful termination, the function passes a return code *EP4GXA_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

C2.12 EP4GXA_Close

EP4GXA_RET EP4GXA_Close (DWORD *inx*);

inx

Defines the card selected from the list of available cards.

Description

The function closes the connection to the card specified with the index *inx*.

The *inx* parameter must match the index of the card to which the process has already opened a connection.

Note: When close a running program, connection between this program and the card is automatically closed (and also if connections with more than one card exists, they are all closed).

Supported cards

This function is implemented for all types of cards supported by the tedia_ep4gxa.dll library.

Return codes

In the case of successful termination, the function passes a return code *EP4GXA_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

C2.13 EP4GXA_BAR0_BaseAdr

```
EP4GXA_RET  EP4GXA_BAR0_BaseAdr    ( DWORD  inx,
                                     size_t  *base_adr,
                                     DWORD  *range );
```

inx

Defines the card selected from the list of available cards.

****base_adr***

Pointer to the memory location where the "BAR0 base address" value was stored by the function.

****range***

Pointer to the memory location where the "BAR0 registers block range" value was stored by the function.

Description

The function provides the "BAR0 base address" and "BAR0 registers block range" values, ie. starting address and block size of the memory for accessing the user function registers.

The *inx* parameter must match the index of the card to which the process has already opened a connection.

Note: An easier and safer way to control the card is to use the functions EP4GXA_ReadReg8, EP4GXA_ReadReg32, EP4GXA_WriteReg8 and EP4GXA_WriteReg32.

Supported cards

This function is implemented for all types of cards supported by the tedia_ep4gxa.dll library.

Return codes

In the case of successful termination, the function passes a return code *EP4GXA_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

C2.14 EP4GXA_ReadReg8, EP4GXA_ReadReg32

```
EP4GXA_RET  EP4GXA_ReadReg8     ( DWORD inx,
                                  DWORD offset,
                                  BYTE *data );
EP4GXA_RET  EP4GXA_ReadReg32    ( DWORD inx,
                                  DWORD offset,
                                  DWORD *data );
```

inx

Defines the card selected from the list of available cards.

offset

Defines the address of the required card register within the memory block with the starting address BAR0 (ie. the address offset against the base address BAR0; this value is described in the card's programming manual).
The entered value must be multiple of 4.

****data***

Pointer to the memory location where the "register data" value was stored by the function.

Description

Functions in variants for 8-bit or 32-bit data are used to reading the contents of the user function registers within BAR0 space; it is necessary for development environments that do not support direct memory access (eg. Visual Basic).
The real address accessed by the function is given by the sum of the values of the base address BAR0 (this value is maintained by the library) and the *offset* values entered by the program. The function therefore allows access to the registers exclusively from the BAR0 space and prevents access outside the valid address range.
The *inx* parameter must match the index of the card to which the process has already opened a connection.

Supported cards

This function is implemented for all types of cards supported by the tedia_ep4gxa.dll library.

Return codes

In the case of successful termination, the function passes a return code *EP4GXA_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

Note: *The execution time of both functions takes approx. 2.5 µs; functions EP4GXA_ReadReg64 or EP4GXA_ReadReg2x32 can be used to achieve higher throughput when reading two adjacent registers.*

C2.15 EP4GXA_WriteReg8, EP4GXA_WriteReg32

```
EP4GXA_RET  EP4GXA_WriteReg8    ( DWORD inx,
                                  DWORD offset,
                                  BYTE data );
EP4GXA_RET  EP4GXA_WriteReg32   ( DWORD inx,
                                  DWORD offset,
                                  DWORD data );
```

inx

Defines the card selected from the list of available cards.

offset

Defines the address of the required card register within the memory block with the starting address BAR0 (ie. the address offset against the base address BAR0; this value is described in the card's programming manual).
The entered value must be multiple of 4.

data

Defines the value written to the card register.

Description

Functions in variants for 8-bit or 32-bit data are used to writting the contents to the user function registers within BAR0 space; it is necessary for development environments that do not support direct memory access (eg. Visual Basic).

The real address accessed by the function is given by the sum of the values of the base address BAR0 (this value is maintained by the library) and the *offset* values entered by the program. The function therefore allows access to the registers exclusively from the BAR0 space and prevents access outside the valid address range.

The *inx* parameter must match the index of the card to which the process has already opened a connection.

Supported cards

This function is implemented for all types of cards supported by the tedia_ep4gxa.dll library.

Return codes

In the case of successful termination, the function passes a return code *EP4GXA_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

Note: *The execution time of both functions takes approx. 2.5 µs; functions EP4GXA_WriteReg64 or EP4GXA_WriteReg2x32 can be used to achieve higher throughput when writing two adjacent registers.*

C2.16 EP4GXA_ReadReg64, EP4GXA_ReadReg2x32 (interface version 3.18 and later)

```
EP4GXA_RET  EP4GXA_ReadReg64    ( DWORD inx,
                                  DWORD offset,
                                  DWORD64  *data );
EP4GXA_RET  EP4GXA_ReadReg2x32  ( DWORD inx,
                                  DWORD offset,
                                  DWORD *dataHi,
                                  DWORD *dataLo );
```

inx

Defines the card selected from the list of available cards.

offset

Defines the address of the required card register within the memory block with the starting address BAR0 (ie. the address offset against the base address BAR0; this value is described in the card's programming manual).
The entered value must be multiple of 8.

****data***

Pointer to the memory location where the "registers data" were stored by the function (64-bit value composed of two 32-bit registers; the lower 32 bits transfer the contents of the register with the address *offset*, the upper 32 bits transfer the contents of the register with the address *offset+4*).

****dataLo***

Pointer to the memory location where the "register data" value (with the address *offset*) was stored by the function.

****dataHi***

Pointer to the memory location where the "register data" value (with the address *offset+4*) was stored by the function.

Description

Functions in variants for one 64-bit are two 32-bit data values is used to reading the contents of the user function registers within BAR0 space; it is necessary for development environments that do not support direct memory access (eg. Visual Basic).

The real address accessed by the function is given by the sum of the values of the base address BAR0 (this value is maintained by the library) and the *offset* values entered by the program. The function therefore allows access to the registers exclusively from the BAR0 space and prevents access outside the valid address range.

The *inx* parameter must match the index of the card to which the process has already opened a connection.

Supported cards

This function is implemented for all types of cards supported by the tedia_ep4gxa.dll library.

Return codes

In the case of successful termination, the function passes a return code *EP4GXA_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

Note: *The execution time of both functions is almost identical to the execution time of the EP4GXA_ReadReg32 function (2,5 µs approx.), so the data throughput is doubled in both 64-bit and 32-bit compiled programs.*

C2.17 EP4GXA_WriteReg64, EP4GXA_WriteReg2x32 (interface version 3.18 and later)

```
EP4GXA_RET  EP4GXA_WriteReg64    ( DWORD inx,
                                   DWORD offset,
                                   DWORD64 data );
EP4GXA_RET  EP4GXA_WriteReg2x32  ( DWORD inx,
                                   DWORD offset,
                                   DWORD dataHi,
                                   DWORD dataLo );
```

inx

Defines the card selected from the list of available cards.

offset

Defines the address of the required card register within the memory block with the starting address BAR0 (ie. the address offset against the base address BAR0; this value is described in the card's programming manual).
The entered value must be multiple of 8.

data

Defines the value written to two card registers (the lower 32 bits transfer the contents of the register with the address *offset*, the upper 32 bits transfer the contents of the register with the address *offset+4*).

dataLo

Defines the value written to the card register with the address *offset*.

dataHi

Defines the value written to the card register with the address *offset+4*.

Description

Functions in variants for one 64-bit or two 32-bit data values are used to writting the contents to the user function registers within BAR0 space; it is necessary for development environments that do not support direct memory access (eg. Visual Basic).

The real address accessed by the function is given by the sum of the values of the base address BAR0 (this value is maintained by the library) and the *offset* values entered by the program. The function therefore allows access to the registers exclusively from the BAR0 space and prevents access outside the valid address range.

The *inx* parameter must match the index of the card to which the process has already opened a connection.

Supported cards

This function is implemented for all types of cards supported by the tedia_ep4gxa.dll library.

Return codes

In the case of successful termination, the function passes a return code *EP4GXA_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

Note: *The execution time of both functions is almost identical to the execution time of the EP4GXA_WriteReg32 function (0,2 µs approx.), so the data throughput is doubled in both 64-bit and 32-bit compiled programs.*

C2.18 EP4GXA_IRQ_Wait

```
EP4GXA_RET  EP4GXA_IRQ_Wait        ( DWORD  inx,
                                     DWORD  * );
```

inx

Defines the card selected from the list of available cards.

***

The second parameter has not been used since interface version 1.55 (maintained for backward compatibility with PCI cards).

Description

The function suspends the thread until the driver detects the interrupt event or until the interrupt timeout period expires (see the description of the *EP4GXA_IRQ_SetTimeout* function).

The *inx* parameter must match the index of the card to which the process has already opened a connection.

Supported cards

This function is implemented for all types of cards supported by the tedia_ep4gxa.dll library.

Return codes

In the case of successful termination, the function passes a return code *EP4GXA_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

The return code *EP4GXA_WaitTimeout* indicates that the function was terminated without errors, but no event caused an interruption of system and function was terminated by the expiration of the preconfigured timeout period.

However, if the function passes the return code *EP4GXA_StopInThread*, it is necessary to stop the execution of the thread because the connection to the card has been probably closed.

C2.19 EP4GXA_IRQ_SetTimeout

```
EP4GXA_RET  EP4GXA_IRQ_SetTimeout   ( DWORD  inx,
                                      DWORD  timeout );
```

inx

Defines the card selected from the list of available cards.

timeout

Defines the interrupt timeout period (ie. latency time) for which the *EP4GXA_IRQ_Wait* function suspends a thread. The value is entered in milliseconds.

Description

This function sets the timeout period (ie. waiting limit for an interrupt event) used by the *EP4GXA_IRQ_Wait* function. If the interrupt event is not detect before the timeout period expires, the *EP4GXA_IRQ_Wait* function is terminated (ie the suspended thread is released) with the return code *EP4GXA_WaitTimeout*.

If the *timeout* value is set to zero, the *EP4GXA_IRQ_Wait* function detects whether an interrupt event has been detected since the *EP4GXA_IRQ_Wait* function was previously executed, and without further waiting, the function is terminated with the return code *EP4GXA_OK* or *EP4GXA_WaitTimeout*.

The *inx* parameter must match the index of the card to which the process has already opened a connection.

Supported cards

This function is implemented for all types of cards supported by the tedia_ep4gxa.dll library.

Return codes

In the case of successful termination, the function passes a return code *EP4GXA_OK* (zero value), otherwise, it passes one of the non-zero error return codes.

C2.20 EP4GXA_IRQ_Count

DWORD EP4GXA_IRQ_Count (DWORD *inx*);

inx

Defines the card selected from the list of available cards.

Description

The function provides the "the number of received interrupts" value for a card specified by the index *inx*.

The *EP4GXA_IRQ_Count* function can also be used if interrupt capability was not enabled when the *EP4GXA_OpenMulti* function connection was opened; the reason is because there may be situations where another program opens (or has already opened) a connection to the card with an enabled interrupt capability.

The *inx* parameter must match the index of the card to which the process has already opened a connection.

Supported cards

This function is implemented for all types of cards supported by the tedia_ep4gxa.dll library.

Return codes

The function does not use the return code method because the returned value directly passes required information.

D1.1 List of standard PCI cards supported by the tedia_ox9162.dll

The PCI cards supported by tedia_ox9162.dll library are equipped with an OX9162 controller (Oxford Semiconductor). Since these cards are compatible only with 5 V signal levels, they are referred to as the first generation of PCI cards.

An overview of the cards is given in the table below.

Standard PCI cards supported by tedia_ox9162.dll		
type	**DID (hex) ***	**description**
PCD-7004	0101	digital ports PCI card
PCD-7104	0102	digital ports PCI card
PCT-7303A	0121	counters & quadrature encoders PCI card
PCT-7408A	0122	counters & timers PCI card
PCT-7424	0123	counters PCI card
PCA-7208AL	0141	multifunction PCI card (AIN, DIO, CNT)
PCA-7208AS	0142	multifunction PCI card (AIN, AOUT, DIO, CNT)
PCA-7408AL	0143	multifunction PCI card (AIN, DIO, CNT)
PCA-7408AS	0144	multifunction PCI card (AIN, AOUT, DIO, CNT)
PCA-7228AL	0145	multifunction PCI card (AIN, DIO, CNT)
PCA-7228AS	0146	multifunction PCI card (AIN, AOUT, DIO, CNT)
PCA-7428AL	0147	multifunction PCI card (AIN, DIO, CNT)
PCA-7428AS	0148	multifunction PCI card (AIN, AOUT, DIO, CNT)
PCA-7228EL	0149	multifunction PCI card (AIN, DIO, CNT)
PCA-7428EL	0150	multifunction PCI card (AIN, DIO, CNT)
PCA-7628AL	0151	multifunction PCI card (AIN, DIO, CNT)
PCA-7628AS	0152	multifunction PCI card (AIN, AOUT, DIO, CNT)
PCA-7288A	0161	analog output PCI card
PCI-1052	0180	special communications PCI card

Note: *The DID column indicates the value returned by the OX9162_Card_DID function (allows to uniquely identify the type of card).*

D2.1 List of standard PCI cards supported by the tedia_ox952.dll

The PCI cards supported by tedia_ox952.dll library are equipped with an OXuPCI952 controller (PLX Technology, formerly Oxford Semiconductor). Since these cards are compatible with both 5 V and 3,3 V signal levels, they are referred to as the second generation of PCI cards.

An overview of the cards is given in the table below.

Standard PCI cards supported by tedia_ox952.dll (with support for both PCI functions) (the function *OX952_BAR05_BaserAdr* provides valid address value)		
type	**DID (hex) ***	**description**
PCA-7428CL	0240	multifunction PCI card (AIN, DIO, CNT)
PCA-7428CS	0242	multifunction PCI card (AIN, AOUT, DIO, CNT)
PCA-7428CE	0244	multifunction PCI card (AIN, DIO, CNT)
PCD-7006C	0302	digital ports PCI card
PCD-7106C	0304	digital ports PCI card
PCT-7303B	0200	counters & quadrature encoders PCI card
PCT-7303C	0210	counters & quadrature encoders PCI card
PCT-7303E	0212	counters & quadrature encoders PCI card
PCT-7424C	0214	counters PCI card
PCT-7424E	0216	counters PCI card
Standard PCI cards supported by tedia_ox952.dll (with support only for PCI function 0) (the function *OX952_BAR05_BaserAdr* cannot be used)		
type	**DID (hex) ***	**description**
PCI-1054U	0401	special communications PCI card

Note: *The DID column indicates the value returned by the OX952_Card_DID function (allows to uniquely identify the type of card).*

D3.1 List of standard PCI Express cards supported by the tedia_ep4gxa.dll

The PCI Express cards supported by tedia_ep4gxa.dll library are equipped with an EP4GX series FPGA (Intel, formerly Altera). The cards are compatible with the PCI Express bus x1 Gen 1.0a/1.1.

An overview of the cards is given in the table below.

Standard PCI Express cards supported by tedia_ep4gxa.dll		
type	**DID (hex) ***	**description**
PCD-8006	0800	digital ports PCIe card
PCD-8104	0804	digital ports PCIe card
PCD-8105	0805	digital ports PCIe card
PCD-8106	0806	digital ports PCIe card
PCT-8303	0810	counters & quadrature encoders PCIe card
PCT-8306	0811	counters & quadrature encoders PCIe card
PCT-8363	0812	counters & quadrature encoders + SSI interface PCIe card
PCT-8360	0820	SSI interface PCIe card
PCT-8424	0830	counters & timers PCIe card
PCT-8425	0831	counters & timers PCIe card
PCT-8426	0832	counters & timers PCIe card
PCA-8428	0840	multifunction PCIe card (AIN, AOUT, DIO, CNT)
PCA-8429	0841	multifunction PCIe card (AIN, DIO, CNT)
PCA-8438	0842	multifunction PCIe card (AIN, AOUT, DIO, CNT)
PCA-8439	0843	multifunction PCIe card (AIN, DIO, CNT)
PCA-8628	0850	multifunction PCIe card (AIN, AOUT, DIO, CNT)
PCA-8629	0851	multifunction PCIe card (AIN, DIO, CNT)
PCA-8638	0852	multifunction PCIe card (AIN, AOUT, DIO, CNT)
PCA-8639	0853	multifunction PCIe card (AIN, DIO, CNT)
PCA-8288	0860	analog output PCIe card (AOUT, DIO)
PCA-8688	0861	analog output PCIe card (AOUT, DIO)
PC8K-LOADER	08FF	any card from the list above programmed with default service firmware (fully functional firmware can be programmed)

Note: *The DID column indicates the value returned by the EP4GXA_Card_DID function (allows to uniquely identify the type of card).*

D4.1 Driver versions and supported Windows operating systems

Driver's version history (from the beginning):

1.x released in 2001
driver built on Jungo Windriver development kit version 5.x
support for PCI cards and 32-bit version of Windows 98/Me, Windows 2000 and Windows XP

2.x released in 2007
driver built on Jungo Windriver development kit version 9.x
support for PCI cards and 32-bit version of Windows 2000, Windows XP, Windows Vista a Windows 7

3.0x released in 2010
driver created as own TEDIA kernel software
support for both 32-bit and 64-bit versions of operating systems
support for Windows 2000, Windows XP, Windows Vista and Windows 7
support for Windows Server 2003, Windows Server 2008 and Windows Server 2008-R2
later added support for Windows 8/8.1 and Windows Server 2012/2012-R2

3.1x released in 2015
added support for PCIe cards
added support for Windows 10 (Windows Server 2016 and higher are not supported)

Driver's version history (from 3.1x upwards):

3.10 available from October 2015
beta version with support for PCIe cards, signed with both SHA-1 and SHA-2 certificates

3.12 available from March 2016
first released version with support for PCIe cards, signed with both SHA-1 and SHA-2 certificates

3.15 available from September 2020
sleep mode, hibernate mode and fast start-up support
support for MSI interrupt mode (PCIe cards only)
signed only with SHA-2 certificates (cannot be used in Windows Vista and Windows Server 2008)

3.18 available from March 2021
several functions for PCIe cards added...
- functions to determine the CardID value (defined by on-board DIP switch position) and the serial number, both functions usable even before opening the connection to the card
- functions for 64-bit accesses to registers usable in both 64-bit and 32-bit compiled programs (provide approx. 2x higher data throughput than 32-bit accesses)

3.20 unreleased version (April 2021)
added functions for DMA transfers memory allocation

the last version with Windows 7/8/8.1 support

3.22 available from May 2022
improved support for Thunderbolt 3/4 interface

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: current information at address
https://www.tedia.eu/contacts

e-mail: current information at addresses
https://www.tedia.eu/contacts
https://www.tedia.eu/support