OPTOIO-PCIe32
ULTRA

EDP-No.: A-840810

32 optocoupler isolated digital inputs
32 optocoupler isolated digital outputs
32*32 bit Counters
Timer
OC and IC Units
Board Identification

wasco®
User's Guide

Registered Trademarks

Windows®, Visual Basic®, Visual C++®, Visual C#® are registered trademarks of Microsoft.
wasco® is registered trademark.
Linux® is registered trademark.
Ubuntu® is registered trademark.
LabVIEW® is registered trademark
Other product and company names mentioned may be trademarks of their respective owners.

Disclaimer

The company Messcomp Datentechnik GmbH assumes no liability for the use of the interface card OPTOIO-PCIe32ULTRA and this documentation, neither for direct nor indirect damages wasco®

Table of Contents

1. Description

OPTOIO-PCIe32ULTRA (board name: WASCO-PCIe8132) provides 32 digital inputs and 32 digital outputs with galvanic isolation, individually for each channel. Optocouplers of high quality ensure the potential separation for the inputs and outputs. Special high-power output optocouplers handle a maximum switching current of up to 150 mA. Each input or output is protected from harmful voltage peaks and pulses by additional protection diodes. You easily can adjust two different input voltage ranges by setting jumpers. A programmable filter can be assigned to each input channel to hide input pulses below an adjustable pulse duration.
In addition to the galvanically isolated inputs and outputs several counters are available as well as Output Compare units (e.g. PWM) and Input Capture units (e.g. for period measurement). All optocoupler inputs, counters, IC units and the two 32-bit timers (time-dependent) can initiate an interrupt. The output optocouplers are connected to a 68-pin SCSI female jack mounted on the board's slot bracket. The optocoupler inputs are connected to a 68-pin onboard SCSI plug. As an option a special plug-in cable set is available, to dislocate the connection to a 68-pin Sub-D jack with slot brackets.
The pin assignment as of the optocoupler inputs and outputs is identical to the PCI cards OPTOIO-PCI32. Therefore a switch to PCIe32ULTRA is easy to implement.
Furthermore, the card provides a jumper block for card identification in order to distinguish several identical cards in your system.

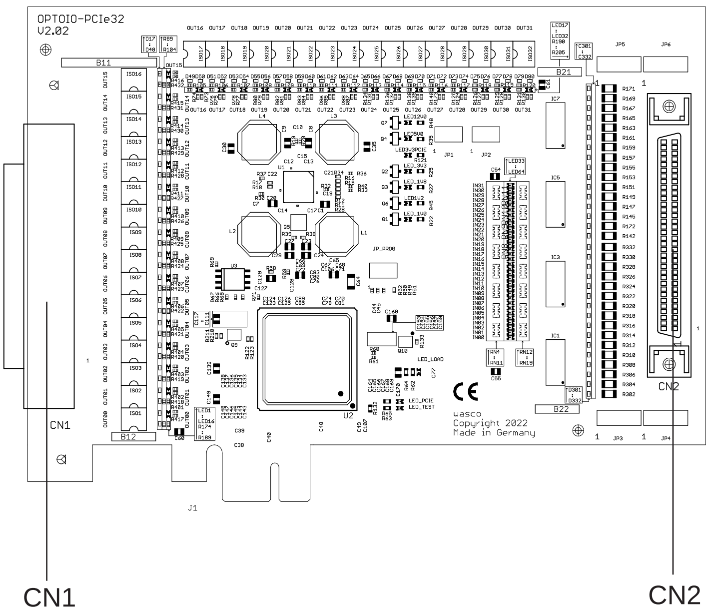
2. Installation of the OPTOIO-PCIe32ULTRA

2.1 Installation of the card into your system

Before you insert the OPTOIO-PCIe32 unplug the power cord or make sure, there is no current to/in the computer. Inserting the interface card in a running system may cause damaging or destroying not only the OPTO-IO-PCIe16ULTRA, but even other already inserted cards of your computer.
Select an empty PCIe slot of your computer for then inserting the card. Please refer to the computer‘s manual for support. Secure the PCB by screwing the slot bracket to the casing of the computer to avoid a card‘s loosening by effects of the connecting cables.

3. Connectors

3.1 Position of the connector plugs on the board



CN1: Optocoupler Outputs OUT00...OUT31
CN2: Optocoupler Inputs IN00...IN31

3.2 Pin assignment of CN1

GND	68	34	Vcc
GND	67	33	Vcc
OUT31-	66	32	OUT31+
OUT30-	65	31	OUT30+
OUT29-	64	30	OUT29+
OUT28-	63	29	OUT28+
OUT27-	62	28	OUT27+
OUT26-	61	27	OUT26+
OUT25-	60	26	OUT25+
OUT24-	59	25	OUT24+
OUT23-	58	24	OUT23+
OUT22-	57	23	OUT22+
OUT21-	56	22	OUT21+
OUT20-	55	21	OUT20+
OUT19-	54	20	OUT19+
OUT18-	53	19	OUT18+
OUT17-	52	18	OUT17+
OUT16-	51	17	OUT16+
OUT15-	50	16	OUT15+
OUT14-	49	15	OUT14+
OUT13-	48	14	OUT13+
OUT12-	47	13	OUT12+
OUT11-	46	12	OUT11+
OUT10-	45	11	OUT10+
OUT09-	44	10	OUT09+
OUT08-	43	9	OUT08+
OUT07-	42	8	OUT07+
OUT06-	41	7	OUT06+
OUT05-	40	6	OUT05+
OUT04-	39	5	OUT04+
OUT03-	38	4	OUT03+
OUT02-	37	3	OUT02+
OUT01-	36	2	OUT01+
OUT00-	35	1	OUT00+

Vcc:
Internal voltage supply (+ 5V) (a wiring bridge must be soldered to B11). **Never apply an external voltage across this pin.**

GND:
Ground connection (only when a wiring bridge is soldered to B12).

3.3 Pin assignment of CN2

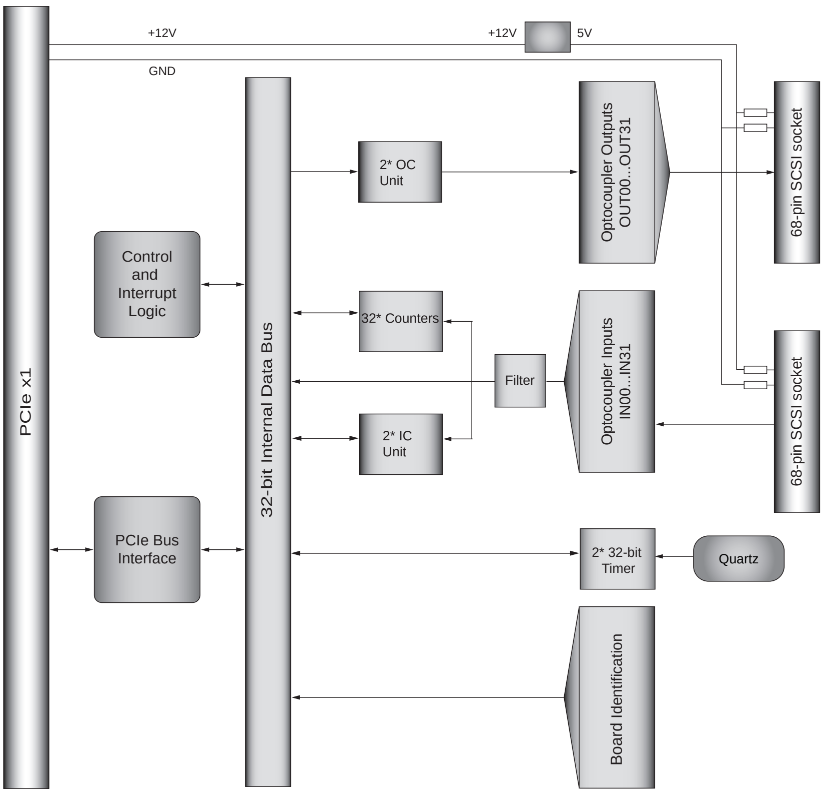
Signal	Pin	Pin	Signal
GND	68	34	Vcc
GND	67	33	Vcc
IN31-	66	32	IN31+
IN30-	65	31	IN30+
IN29-	64	30	IN29+
IN28-	63	29	IN28+
IN27-	62	28	IN27+
IN26-	61	27	IN26+
IN25-	60	26	IN25+
IN24-	59	25	IN24+
IN23-	58	24	IN23+
IN22-	57	23	IN22+
IN21-	56	22	IN21+
IN20-	55	21	IN20+
IN19-	54	20	IN19+
IN18-	53	19	IN18+
IN17-	52	18	IN17+
IN16-	51	17	IN16+
IN15-	50	16	IN15+
IN14-	49	15	IN14+
IN13-	48	14	IN13+
IN12-	47	13	IN12+
IN11-	46	12	IN11+
IN10-	45	11	IN10+
IN09-	44	10	IN09+
IN08-	43	9	IN08+
IN07-	42	8	IN07+
IN06-	41	7	IN06+
IN05-	40	6	IN05+
IN04-	39	5	IN04+
IN03-	38	4	IN03+
IN02-	37	3	IN02+
IN01-	36	2	IN01+
IN00-	35	1	IN00+

Vcc:
Internal voltage supply (+ 5V) (a wiring bridge must be soldered to B21). **Never apply an external voltage across this pin.**

GND:
Ground connection (only when a wiring bridge is soldered to B22).

4. System Components

4.1 Block Diagram



4.2 Access to the system components

You can access to the hardware components of the OPTOIO-PCIe32 by reading from or writing to Memory Mapped I/O addresses using library functions. The addresses relevant to the OPTOIO-PCIe32 arise depending on the BIOS assigned base address. Access to the OPTOIO-PCIe32 is exclusively in double-word access (you will find more information in the chapter Programming as well as in the example programs on the supplied CD)

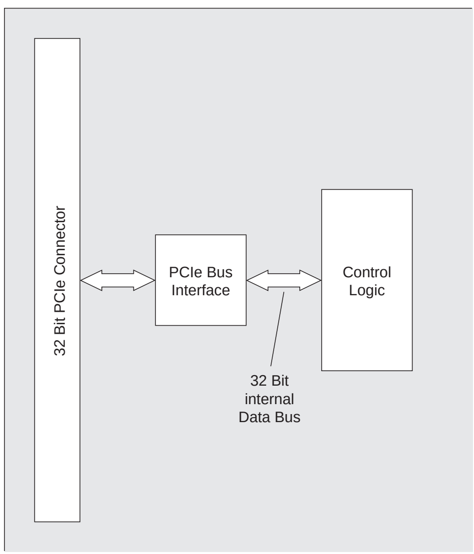


Fig 4.2

5. 32 Optocoupler Isolated Digital Inputs

The OPTOIO-PCIe32 provides 32 input channels, each of which is optically isolated by optocouplers. The isolation voltage between GND and input is 500 V_{DC}. The voltage within the input channels is limited to 50 V_{DC}.

5.1 Pin assignment of the input optocouplers

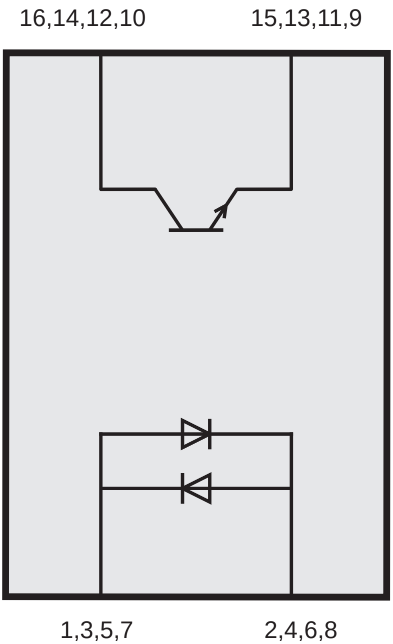


Fig. 5.1

5.2 Input voltage ranges

You can choose between two different input voltage ranges for each opto-coupler input by setting jumpers on the blocks JP3, JP4, JP5 and JP6.

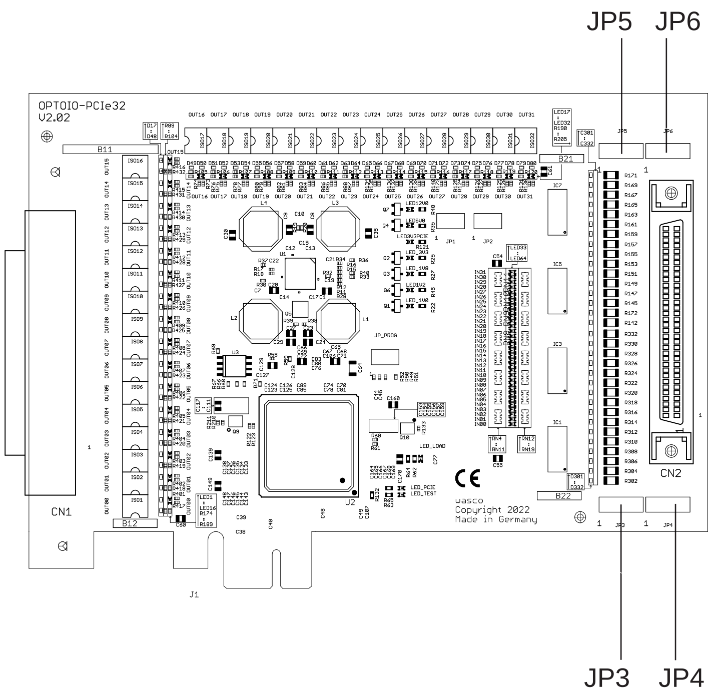
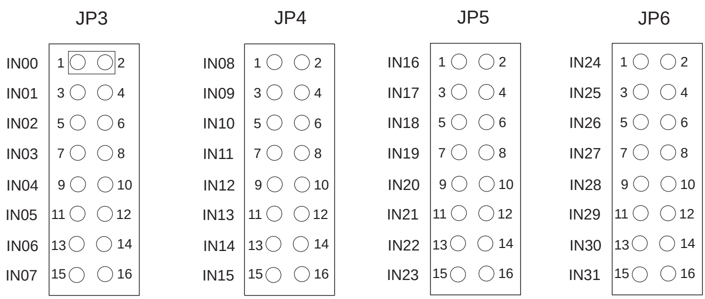


Fig. 5.2

For the data of the two input voltage ranges, please refer to the following table:

Jumper	**LOW**	**HIGH**
closed	0...1 V	5...15 V
open	0...2 V	14...30 V



Example:
By placing a jumper over Pin1 and Pin2 of the jumper block JP3 the input voltage range of IN00 changes from 0..2V (Low) and 14..30V (High) to 0..1V (Low) and 5..15V (High). The remaining input voltage ranges keep unaffected.

5.3 Input wiring

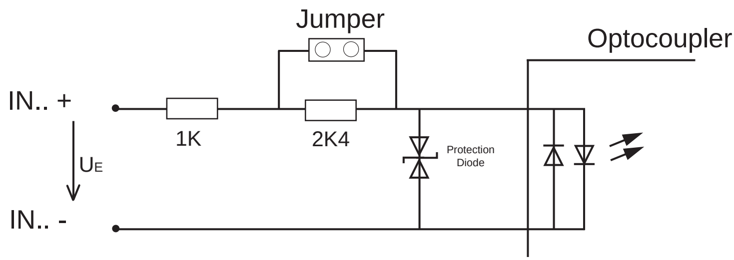


Fig 5.3

5.4 Input current

$$I_E \approx \frac{U_E - 1{,}1V}{3400\Omega} \qquad \text{(Jumper open)}$$

$$I_E \approx \frac{U_E - 1{,}1V}{1000\Omega} \qquad \text{(Jumper closed)}$$

5.5 Access to the inputs

In order to determine the state of the optocoupler inputs, the register OPTOIN has to be read out. Every bit of the 32-bit value stands for one input as shown in the register table.

Application example:
As an example every third optocoupler input of the input plug shall be set to HIGH and all the rest to LOW. When the register OPTOIN is read, the card returns the value 0x44444444(hex). Out of this value you can filter the states of every single input by an AND-link operation.

5.6 Optocoupler inputs with digital filters

Each of the optocoupler inputs of the board WASCO-PCIe8132 has its own configurable digital filter to filter spurious pulses and transients of the input signal.
For that the filter checks whether or not a signal is applied long enough, as shown in fig. 5.5. If this is not the case, a too short pulse for example will be ignored. In register OPTOINFILx you can adjust the minimum time of how long the signal has to be applied to be considered. You can adjust a filter width of 0 - 65535µs in steps of 1µs.
In state of default the filter is deactivated, say the filter duration is 0 µs.

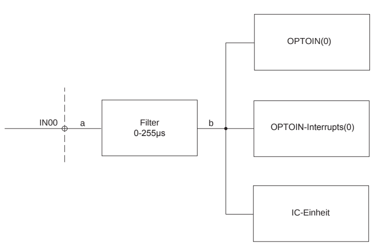


Fig. 5.4

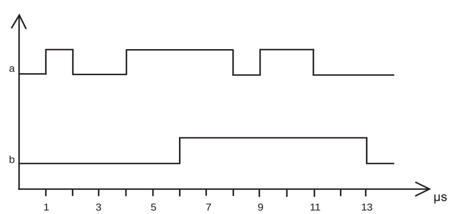


Fig. 5.5

Using the filters please note that the optocoupler inputs on one hand have response times much longer than 1µs and on the other hand may vary in steepness of rising and falling curves. This is not considered at the filter configuration, as the optocouplers may show varying component tolerances. The user has to allow for this by himself. Not considerung these switching times may lead to a filtering out of signals at the optocoupler input, even though theoretically they are applied long enough.

5.7 Interrupt functions of the optocoupler inputs

To detect changes to the optocoupler inputs without periodically querying the input state via PC, the OPTOIO-PCIe32 offers several interrupt options.
On one hand the card is aible to trigger an interrupt on one of the inputs on a rising edge. On the other hand the card can signal to the PC a general change of the input states by an interrupt.
For further information please see the chapter Interrupt Controller.

5.7.1 Edge detection

In order to detect rising edges at the optocoupler inputs, each single input provides an edge detection with connectable interrupt function. For this purpose, a 32-bit interrupt register (OPTOINIF) is provided which makes available one bit per input channel for edge detection. As soon as the card detects a rising edge, the respective bit is set in register OPTOINIF. If at least one of the enabled bits is set, this will be passed over a line to the interrupt controller.

The interrupt function is enabled by writing the 32bit register OPTOINFe. Each single bit represents one input. As shown in the table port addresses (chapter 11.1) the respective bit indicates activation of the interrupt function with a 1 and deactivation with a 0. So, if the bit is 0, the corresponding bit in the register OPTOINIF will be set on a rising edge, but it will not be considered when the interrupt is triggered.
All of the interrupt channels are deactivated in default state.

After the interrupt has been triggered, the source must be determined in the relevant interrupt service routine by reading the register OPTOINIF. Then the bit has to be cleared by setting the source channel bit in the register OPTOINFr. After the card has executed the reset command, the bit is reset automatically.

Application example:
You want an edge detection with interrupt triggering on channel IN01. The following example lists each step of how to perform the configuration and what needs to be done in the interrupt service routine to re-enable the interrupt.
Please note that in this example the interrupt configuration of the driver is not indicated. For the discription of this please refer to the driver.
Additional, more program examples are made available for download on our homepage.

Configuration:

1. Activation of the card‘s interrupt function (see chapter Interrupt Controller)
2. Enable the required interrupt
Before enabling an edge detection interrupt, please check whether or not the edge memory register OPTOINIF is reset completely. Otherwise, an interrupt might be triggered immediately after enabling the interrupt. If not all of the bits are reset in register OPTOINIF, write the value 0xffffffff(hex) into the register OPTOINIFr.
See the table how to set bit 1 in register OPTOINIFe to activate an edge detection interrupt on channel IN01. This way, with the help of the PCIe write command, the value 0x00000002(hex) resp. 2(dec) is written to this register.

Interrupt Service Routine

1. To determine the interrupt source, the edge memory register OPTOINIF has to be read out (return value here 0x00000002(hex). If other sources are available, such as timer etc., please check in INTCON register whether or not the interrupt received from the PC is derived from the OPTOINIF register.

2. Once the source is identified, the source bit must be cleared. For this purpose, in our case write the value 0x00000002(hex) to the register OPTOINIFr.

Attention:
If in that time further interrupts were triggered (e.g. Timer), these must be deleted in their respective registers, too. Only after all of the activated interrupt registers have been reset to 0 again, another interrupt can be triggered.

5.7.2 Port Changes

If the optocoupler inputs often need to be queried to detect changes, another interrupt function can be used to relieve the PC. For this the WASCO-PCIe8132 provides the possibiltiy to trigger an interrupt in the event of a change at the inputs.
To enable this interrupt function on one hand the register OPTOINICe has to be set to 0x00000001. On the other hand, the user can determine via the 32-bit register OPTOINICCe, which one of the inputs should be considered for the detection. In the event of a change at the inputs, the corresponding bit is set in register OPTOINIC. To re-enable the interrupt after having been triggered, the corresponding bit in register OPTOINICr has to be set.
After the reset, the reset bit will be reset automatically.

5.8 Port Addresses

Offset-Address	Register Name	Bit Range	Bits															
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x0000	OPTOIN	31:16	OPTOIN <31:16>															
		15:0	OPTOIN <15:0>															
0x00A0	OPTOINFIL0	31:16	reserved (*)															
		15:0	reserved (*)								OPTOINFIL0 <7:0>							
0x00A4	OPTOINFIL1	31:16	reserved (*)															
		15:0	reserved (*)								OPTOINFIL1 <7:0>							
0x00A8	OPTOINFIL2	31:16	reserved (*)															
		15:0	reserved (*)								OPTOINFIL2 <7:0>							
0x00AC	OPTOINFIL3	31:16	reserved (*)															
		15:0	reserved (*)								OPTOINFIL3 <7:0>							
0x00B0	OPTOINFIL4	31:16	reserved (*)															
		15:0	reserved (*)								OPTOINFIL4 <7:0>							
0x00B4	OPTOINFIL5	31:16	reserved (*)															
		15:0	reserved (*)								OPTOINFIL5 <7:0>							
0x00C0	OPTOINFIL6	31:16	reserved (*)															
		15:0	reserved (*)								OPTOINFIL6 <7:0>							
0x00BC	OPTOINFIL7	31:16	reserved (*)															
		15:0	reserved (*)								OPTOINFIL7 <7:0>							
0x00C0	OPTOINFIL8	31:16	reserved (*)															
		15:0	reserved (*)								OPTOINFIL8 <7:0>							
0x00C4	OPTOINFIL9	31:16	reserved (*)															
		15:0	reserved (*)								OPTOINFIL9 <7:0>							
0x00C8	OPTOINFIL10	31:16	reserved (*)															
		15:0	reserved (*)								OPTOINFIL10 <7:0>							

Offset-Address	Register Name	Bit Range	Bits															
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x00CC	OPTOINFIL11	31:16	reserved (*)															
		15:0	OPTOINFIL11 <15:0>															
0x00D0	OPTOINFIL12	31:16	reserved (*)															
		15:0	OPTOINFIL12 <15:0>															
0x00D4	OPTOINFIL13	31:16	reserved (*)															
		15:0	OPTOINFIL13 <15:0>															
0x00D8	OPTOINFIL14	31:16	reserved (*)															
		15:0	OPTOINFIL14 <15:0>															
0x00DC	OPTOINFIL15	31:16	reserved (*)															
		15:0	OPTOINFIL15 <15:0>															
0x00E0	OPTOINFIL16	31:16	reserved (*)															
		15:0	OPTOINFIL16 <15:0>															
0x00E4	OPTOINFIL17	31:16	reserved (*)															
		15:0	OPTOINFIL17 <15:0>															
0x00E8	OPTOINFIL18	31:16	reserved (*)															
		15:0	OPTOINFIL18 <15:0>															
0x00EC	OPTOINFIL19	31:16	reserved (*)															
		15:0	OPTOINFIL19 <15:0>															
0x00F0	OPTOINFIL20	31:16	reserved (*)															
		15:0	OPTOINFIL20 <15:0>															
0x00F4	OPTOINFIL21	31:16	reserved (*)															
		15:0	OPTOINFIL21 <15:0>															
0x00F8	OPTOINFIL22	31:16	reserved (*)															
		15:0	OPTOINFIL22 <15:0>															

(*) reserved area has to be assigned with 0

Offset-Address	Register Name	Bit Range	Bits															
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x00FC	OPTOINFIL23	31:16	reserved (*)															
		15:0	OPTOINFIL23 <15:0>															
0x0100	OPTOINFIL24	31:16	reserved (*)															
		15:0	OPTOINFIL24 <15:0>															
0x0104	OPTOINFIL25	31:16	reserved (*)															
		15:0	OPTOINFIL25 <15:0>															
0x0108	OPTOINFIL26	31:16	reserved (*)															
		15:0	OPTOINFIL26 <15:0>															
0x010C	OPTOINFIL27	31:16	reserved (*)															
		15:0	OPTOINFIL27 <15:0>															
0x0110	OPTOINFIL28	31:16	reserved (*)															
		15:0	OPTOINFIL28 <15:0>															
0x0114	OPTOINFIL29	31:16	reserved (*)															
		15:0	OPTOINFIL29 <15:0>															
0x0118	OPTOINFIL30	31:16	reserved (*)															
		15:0	OPTOINFIL30 <15:0>															
0x011C	OPTOINFIL31	31:16	reserved (*)															
		15:0	OPTOINFIL31 <15:0>															

(*) reserved area has to be assigned with 0

Register OPTOIN:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R							
	OPTOIN <31:24>							
23:16	R							
	OPTOIN <23:16>							
15:8	R							
	OPTOIN <15:8>							
7:0	R							
	OPTOIN < 7:0>							

Bit 31 - 0 **OPTOIN <31:0>**
each bit corresponds to one optocoupler input
(e.g. IN00 = OPTOIN<0>, IN13 = OPTOIN<13>)
If a HIGH is applied to an input, the associated bit is 1, otherwise it is 0

Register OPTOINFILx:

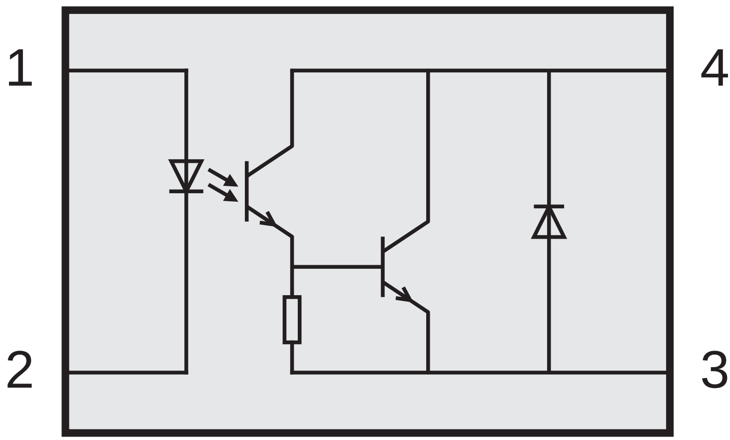
Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W							
	-							
23:16	R/W							
	-							
15:8	R/W							
	OPTOINFILx < 15:8>							
7:0	R/W							
	OPTOINFILx < 7:0>							

Bit 31 - 16 reserved (value 0 is written)
Bit 15 - 0 **OPTOINFILx <15:0>** (default = 0)
This value determines the filter duration of the filter x in µs

6. 32 Optocoupler Outputs

The OPTOIO-PCIe32 provides 32 output channels, each of which is optically isolated by optocouplers as well. The isolation voltage between GND and output is 500 V.

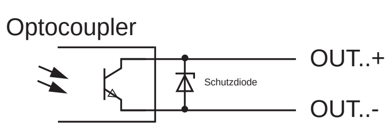
6.1 Pin assignment of the output optocouplers



6.2 Optocoupler data

Voltage CE:	max. 50V
Voltage EC:	0,1V
Current CE:	150 mA

6.3 Output wiring



6.4 Functions of the optocoupler outputs

6.4.1 Basic function

The basic function of the optocoupler outputs allows the locking or enabling of the single outputs by writing to the 32-bit register OPTOOUT. In this register every single bit stands for one optocoupler output, as shown in table Port Addresses.
For example, if you want to connect every third output of the connector, you have to write the value 0x44444444(hex) to the register OPTOOUT.

6.4.2 Assigning optocouplers with other hardware components

In addition to the basic function, which allows easy access to the optocoupler outputs, it is possible to assign different hardware components to the individual outputs, such as a PWM output (see Fig. 6.4.1). For this purpose, every optocoupler has a multiplexer with a 4-bit addressing (= up to 16 different sources). As a default source, the register OPTOOUT is specified as peripheral after a reset or when booting the PC. To change the source, the source address (see Fig. 6.4.2) has to be written to the register OPTOOUTMUXx

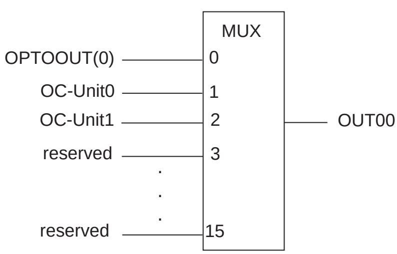


Fig. 6.4.1

Address	Peripheral
0x0 (default)	OPTOOUT(x)
0x1	OC-Unit0
0x2	OC-Unit1
0x3 - 0xF	reserved

Fig. 6.4.2

6.5 Port Addresses

Offset-Address	Register Name	Bit Range	Bits															
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x0008	OPTOOUT	31:16	reserved (*)															
		15:0	OPTOOUT <15:0>															
0x03C0	OPTOOUTMUX0	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX0 [3:0]			
0x03C4	OPTOOUTMUX1	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX1 [3:0]			
0x03C8	OPTOOUTMUX2	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX2 [3:0]			
0x03CC	OPTOOUTMUX3	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX3 [3:0]			
0x03D0	OPTOOUTMUX4	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX4 [3:0]			
0x03D4	OPTOOUTMUX5	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX5 [3:0]			
0x03D8	OPTOOUTMUX6	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX6 [3:0]			
0x03DC	OPTOOUTMUX7	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX7 [3:0]			
0x03E0	OPTOOUTMUX8	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX8 [3:0]			
0x03E4	OPTOOUTMUX9	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX9 [3:0]			
0x03E8	OPTOOUTMUX10	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX10 [3:0]			

(*) reserved area has to be assigned with 0

Offset-Address	Register Name	Bit Range	Bits															
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x03EC	OPTOOUTMUX11	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX11 [3:0]			
0x03F0	OPTOOUTMUX12	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX12 [3:0]			
0x03F4	OPTOOUTMUX13	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX13 [3:0]			
0x03F8	OPTOOUTMUX14	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX14 [3:0]			
0x03FC	OPTOOUTMUX15	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX15 [3:0]			
0x0400	OPTOOUTMUX16	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX16 [3:0]			
0x0404	OPTOOUTMUX17	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX17 [3:0]			
0x0408	OPTOOUTMUX18	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX18 [3:0]			
0x040C	OPTOOUTMUX19	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX19 [3:0]			
0x0410	OPTOOUTMUX20	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX20 [3:0]			
0x0414	OPTOOUTMUX19	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX21 [3:0]			
0x0418	OPTOOUTMUX20	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX22 [3:0]			

(*) reserved area has to be assigned with 0

Offset-Address	Register Name	Bit Range	Bits 31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x041C	OPTOOUTMUX23	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX23 [3:0]			
0x0420	OPTOOUTMUX24	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX24 [3:0]			
0x0424	OPTOOUTMUX25	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX25 [3:0]			
0x0428	OPTOOUTMUX26	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX26 [3:0]			
0x042C	OPTOOUTMUX27	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX27 [3:0]			
0x0430	OPTOOUTMUX28	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX28 [3:0]			
0x0434	OPTOOUTMUX29	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX29 [3:0]			
0x0438	OPTOOUTMUX30	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX30 [3:0]			
0x043C	OPTOOUTMUX31	31:16	reserved (*)															
		15:0	reserved (*)												OPTOOUTMUX31 [3:0]			

(*) reserved area has to be assigned with 0

Register OPTOOUT:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W							
	OPTOOUT <31:24>							
23:16	R/W							
	OPTOOUT <23:16>							
15:8	R/W							
	OPTOOUT <15:8>							
7:0	R/W							
	OPTOOUT <7:0>							

Bit 31 - 0 **OPTOOUT <31:0>** (default = 0)
The value determines the state of the output optocouplers.
Each bit corresponds to one optocoupler output
(OPTOOUT<0> = OUT00, OPTOOUT<13> = OUT13)
If the respective bit is 1, the corresponding optocoupler output is enabled. If the respective bit is 0, the corresponding optocoupler output is locked.

Register OPTOOUTMUXx:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U							
	reserved							
23:16	U							
	reserved							
15:8	U							
	reserved							
7:0	U				R/W			
	reserved				OPTOOUTMUXx <3:0>			

Bit 31 - 4 reserved (value 0 is written)
Bit 3 - 0 **OPTOOUTMUXx <3:0>** (default = 0)
Determines, which peripheral is connected to the output optocoupler
0 = register OPTOOUT
1 = OC-Unit0
2 = OC-Unit1
3 - 15 = reserved

7. Counter

The board WASCO-PCIe8132 provides a total of 32 32-bit event counters (rising edges). Every single counter can be freely assigned to one digital input. Furthermore, each counter can trigger an interrupt in the event of an overflow.

7.1 Basic function

1. For using a counter start by selecting a source. For this purpose every counter has its own 32-bit register (COUNTMUXxx).
2. Next, the counter has to be preloaded via the register COUNTLDxxx. In general, the value 0 is written to the register.
3. Finally, the counter is activated by setting the first bit in the register COUNTexx. From this point, the counter starts to count every rising edge. In the event of an overflow the bit corresponding to the counter is set in the register COUNTIR. To detect another overflow, this bit has to be cleared by setting the bit allocated to the counter in the register COUNTIRr.
4. To determine the counter value read out the register COUNTxx.

7.2 Interrupt function

Every overflow of a counter sets the bit allocated to the counter in the register COUNTIR. When the interrupt line has been enabled by setting the relevant bit in the register COUNTIRe, then the overflow will be passed on to the interrupt controller. In order to be able to reset the overflow bit, the bit assigned to the counter has to be set in the register COUNTIRr. After an internal reset of an overflow bit, the bit set in the register COUNTIRr will be reset automatically.

7.3 Port Addresses

Offset-Address	Register Name	Bit Range	Bits 31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x1000	COUNT0e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1004	COUNT1e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1008	COUNT2e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x100C	COUNT3e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1010	COUNT4e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1014	COUNT5e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1018	COUNT6e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x101C	COUNT7e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1020	COUNT8e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1024	COUNT9e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1028	COUNT10e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x102C	COUNT11e	31:16	reserved (*)															
		15:0	reserved (*)															en

(*) reserved area has to be assigned with 0

Offset-Address	Register Name	Bit Range	Bits															
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x1030	COUNT12e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1034	COUNT13e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1038	COUNT14e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x103C	COUNT15e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1040	COUNT16e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1044	COUNT17e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1048	COUNT18e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x104C	COUNT19e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1050	COUNT20e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1054	COUNT21e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1058	COUNT22e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x105C	COUNT23e	31:16	reserved (*)															
		15:0	reserved (*)															en

(*) reserved area has to be assigned with 0

Offset-Address	Register Name	Bit Range	Bits															
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x1060	COUNT24e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1064	COUNT25e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1068	COUNT26e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x106C	COUNT27e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1070	COUNT28e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1074	COUNT29e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1078	COUNT30e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x107C	COUNT31e	31:16	reserved (*)															
		15:0	reserved (*)															en

(*) reserved area has to be assigned with 0

Offset-Address	Register Name	Bit Range	Bits															
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x1100	COUNT0	31:16	COUNT0<31:16>															
		15:0	COUNT0<15:0>															
0x1104	COUNT1	31:16	COUNT1<31:16>															
		15:0	COUNT1<15:0>															
0x1108	COUNT2	31:16	COUNT2<31:16>															
		15:0	COUNT2<15:0>															
0x110C	COUNT3	31:16	COUNT3<31:16>															
		15:0	COUNT3<15:0>															
0x1110	COUNT4	31:16	COUNT4<31:16>															
		15:0	COUNT4<15:0>															
0x1114	COUNT5	31:16	COUNT5<31:16>															
		15:0	COUNT5<15:0>															
0x1118	COUNT6	31:16	COUNT6<31:16>															
		15:0	COUNT6<15:0>															
0x111C	COUNT7	31:16	COUNT7<31:16>															
		15:0	COUNT7<15:0>															
0x1120	COUNT8	31:16	COUNT8<31:16>															
		15:0	COUNT8<15:0>															
0x1124	COUNT9	31:16	COUNT9<31:16>															
		15:0	COUNT9<15:0>															
0x1128	COUNT10	31:16	COUNT10<31:16>															
		15:0	COUNT10<15:0>															

(*) reserved area has to be assigned with 0

Offset-Address	Register Name	Bit Range	Bits															
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x112C	COUNT11	31:16	COUNT11<31:16>															
		15:0	COUNT11<15:0>															
0x1130	COUNT12	31:16	COUNT12<31:16>															
		15:0	COUNT12<15:0>															
0x1134	COUNT13	31:16	COUNT13<31:16>															
		15:0	COUNT13<15:0>															
0x1138	COUNT14	31:16	COUNT14<31:16>															
		15:0	COUNT14<15:0>															
0x113C	COUNT15	31:16	COUNT15<31:16>															
		15:0	COUNT15<15:0>															
0x1140	COUNT16	31:16	COUNT16<31:16>															
		15:0	COUNT16<15:0>															
0x1144	COUNT17	31:16	COUNT17<31:16>															
		15:0	COUNT17<15:0>															
0x1148	COUNT18	31:16	COUNT18<31:16>															
		15:0	COUNT18<15:0>															
0x114C	COUNT19	31:16	COUNT19<31:16>															
		15:0	COUNT19<15:0>															
0x1150	COUNT20	31:16	COUNT20<31:16>															
		15:0	COUNT20<15:0>															
0x1154	COUNT21	31:16	COUNT21<31:16>															
		15:0	COUNT21<15:0>															
0x1158	COUNT22	31:16	COUNT22<31:16>															
		15:0	COUNT22<15:0>															

(*) reserved area has to be assigned with 0

Offset-Address	Register Name	Bit Range	Bits															
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x115C	COUNT23	31:16	COUNT23<31:16>															
		15:0	COUNT23<15:0>															
0x1160	COUNT24	31:16	COUNT24<31:16>															
		15:0	COUNT24<15:0>															
0x1164	COUNT25	31:16	COUNT25<31:16>															
		15:0	COUNT25<15:0>															
0x1168	COUNT26	31:16	COUNT26<31:16>															
		15:0	COUNT26<15:0>															
0x116C	COUNT27	31:16	COUNT27<31:16>															
		15:0	COUNT27<15:0>															
0x1170	COUNT28	31:16	COUNT28<31:16>															
		15:0	COUNT28<15:0>															
0x1174	COUNT29	31:16	COUNT29<31:16>															
		15:0	COUNT29<15:0>															
0x1178	COUNT30	31:16	COUNT30<31:16>															
		15:0	COUNT30<15:0>															
0x117C	COUNT31	31:16	COUNT31<31:16>															
		15:0	COUNT31<15:0>															
0x1180	COUNTMUX0	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX0 <7:0>							
0x1184	COUNTMUX1	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX1 <7:0>							
0x1188	COUNTMUX2	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX2 <7:0>							

(*) reserved area has to be assigned with 0

Offset-Address	Register Name	Bit Range	Bits															
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x118C	COUNTMUX3	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX3 <7:0>							
0x1190	COUNTMUX4	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX4 <7:0>							
0x1194	COUNTMUX5	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX5 <7:0>							
0x1198	COUNTMUX6	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX6 <7:0>							
0x119C	COUNTMUX7	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX7 <7:0>							
0x11A0	COUNTMUX8	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX8 <7:0>							
0x11A4	COUNTMUX9	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX9 <7:0>							
0x11A8	COUNTMUX10	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX10 <7:0>							
0x11AC	COUNTMUX11	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX11 <7:0>							
0x11B0	COUNTMUX12	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX12 <7:0>							
0x11B4	COUNTMUX13	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX13 <7:0>							
0x11B8	COUNTMUX14	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX14 <7:0>							

(*) reserved area has to be assigned with 0

Offset-Address	Register Name	Bit Range	Bits															
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x118C	COUNTMUX15	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX15 <7:0>							
0x1190	COUNTMUX16	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX16 <7:0>							
0x1194	COUNTMUX17	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX17 <7:0>							
0x1198	COUNTMUX18	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX18 <7:0>							
0x119C	COUNTMUX19	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX19 <7:0>							
0x11A0	COUNTMUX20	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX20 <7:0>							
0x11A4	COUNTMUX21	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX21 <7:0>							
0x11A8	COUNTMUX22	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX22 <7:0>							
0x11AC	COUNTMUX23	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX23 <7:0>							
0x11B0	COUNTMUX24	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX24 <7:0>							
0x11B4	COUNTMUX25	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX25 <7:0>							
0x11B8	COUNTMUX26	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX26 <7:0>							

(*) reserved area has to be assigned with 0

Offset-Address	Register Name	Bit Range	Bits															
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x11BC	COUNTMUX27	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX27 <7:0>							
0x11C0	COUNTMUX28	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX28 <7:0>							
0x11C4	COUNTMUX29	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX29 <7:0>							
0x11C8	COUNTMUX30	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX30 <7:0>							
0x11CC	COUNTMUX31	31:16	reserved (*)															
		15:0	reserved (*)								COUNTMUX31 <7:0>							

(*) reserved area has to be assigned with 0

Register COUNTxe:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U							
	reserved							
23:16	U							
	reserved							
15:8	U							
	reserved							
7:0	U							R/W
	reserved							en

Bit 31 - 1 reserved (value 0 is written)

Bit 0 **COUNTxe<0>** (default = 0)
lock or enable the counter
0 = lock (default)
1 = enable

Register COUNTx:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W							
	COUNTx <31:24>							
23:16	R/W							
	COUNTx <23:16>							
15:8	R/W							
	COUNTx <315:8>							
7:0	R/W							
	COUNTx <7:0>							

Bit 31 - 0 **COUNTx <31:0>** (default = 0)
this register allows to read out the current counter value of the counter x and to write to (for example for the initial state).

Register COUNTMUXx:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U							
	reserved							
23:16	U							
	reserved							
15:8	U							
	reserved							
7:0	R/W							
	COUNTMUXx <7:0>							

Bit 31 - 8 reserved (value 0 is written)

Bit 7 - 0 **COUNTMUXx <7:0>** (default = 0)

the register value determines the card peripheral applied to the counter

0 = Optocoupler input IN00 (default)

1 = Optocoupler input IN01

.

.

.

31 = Optocoupler input IN31

255 - 32 = reserved -> to be assigned with 0

8. Timer

The available 32-bit timers can be used as a timer or for configurable interval interrupt triggering. For this, intervals between 0 and 4294967295µs can be adjusted in steps of 1µs.

8.1 Using as an interval interrupt trigger

1. Start with clearing the timer x by deleting the bit 0 of the register TIMERxe and then reset the timer. This reset is executed by writing the value 0 to the register TIMERx.
2. Next, determinate the interval. The duration of the interval is set in the writable 32-bit register TIMERCOMPx
Interval duration = (TIMERCOMPx + 1)*1µs
3. In order to trigger an interrupt after the interval has elapsed, the timer has to be enabled. For this, set the corresponding bit in register TIMERIRe. (Attention: the interrupt controller has to be enabled, too)
4. The timer being configured completely, activate it by setting bit 0 in the register TIMERxe.
5. If the interrupt has been triggered, this can be checked in the register TIMERIR. To receive a new interrupt, the source bit must be cleared by setting the respective reset bit in register TIMERIRr

8.2 Port Addresses

Offset-Address	Register Name	Bit Range	31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x1400	TIMER0e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1404	TIMER1e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1420	TIMER0	31:16	TIMER0 <31:16>															
		15:0	TIMER0 <15:0>															
0x1424	TIMER1	31:16	TIMER1 <31:16>															
		15:0	TIMER0 <15:0>															
0x1430	TIMERCOMP0	31:16	TIMERCOMP0 <31:16>															
		15:0	TIMERCOMP0 <15:0>															
0x1434	TIMERCOMP1	31:16	TIMERCOMP1 <31:16>															
		15:0	TIMERCOMP1 <15:0>															

(*) reserved area has to be assigned with 0

Register TIMERxe:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U							
	reserved							
23:16	U							
	reserved							
15:8	U							
	reserved							
7:0	U							R/W
	reserved							en

Bit 31 - 1 reserved (value 0 is written)

Bit 0 **TIMERxe<0>** (default = 0)
start or stop the timer
0 = stopped (default)
1 = started

Register TIMERx:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W							
	TIMERx<31:24>							
23:16	R/W							
	TIMERx<23:16>							
15:8	R/W							
	TIMERx<15:8>							
7:0	R/W							
	TIMERx<7:0>							

Bit 31 - 0 **TIMERx<0>** (default = 0)
this register allows to read out the current value of the timer x and to write to (for example for the initial value).

Register TIMERCOMPx:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W							
	TIMERCOMPx<31:24>							
23:16	R/W							
	TIMERCOMPx<23:16>							
15:8	R/W							
	TIMERCOMPx<15:8>							
7:0	R/W							
	TIMERCOMPx<7:0>							

Bit 31 - 0 **TIMERCOMPx<0>** (default = 0)
The value of the register TIMERCOMP determines the interval duration of the timer
TIMERCOMP = Interval duration - 1

9. Input Capture Unit

The Input Capture Units (IC-Unit) allow to measure pulse duration and period of received signals. Each one of the units has its own 32-bit timer for time measurement in steps of 1µs, and can be assigned to any digital input by programming.

9.1 Continuous measurement of periodic signals

In this mode, the input signal is scanned regularly when the function is activated, and the period duration and pulse duration is determined. For this, the unit starts measuring at the first rising edge at the input and ends it at the following rising edge. Measurement completed, automatically the period duration and the pulse duration is computed and the values are written to the registers ICPERIODLx and ICPULSLx. At the next rising edge, the unit starts to measure by itself.

9.1.1 Application

1. Make sure, the intended unit to be deactivated before configuration. The IC Unit is disabled by clearing the bit 0 in the register ICUNITex.
2. When the IC-Unit is deactivated, carry out the configuration in register ICCONFIGx. For the continuous measurement of periodic signals write the value b0000(bin) in the mode section.
3. When the unit is configured, then the source has to be selected by writing it into register ICMUXx.
4. Now to start the measurement, set bit 0 in the register ICUNITex.

Attention: please pay attention to the varying switching delays when using the optocoupler inputs. These change the pulse width.

9.1.2 Interrupt function

In addition to the measurement of the period and the pulse duration, it is possible to trigger an interrupt after completion. For this you activate the interrupt function by setting the corresponding bit in register ICUNITRe. When the interrupt is triggered, read out the source in register ICUNITIR and again activate the source by setting the corresponding bit in register ICUNITIRr.

9.2 Port Addresses

Offset-Address	Register Name	Bit Range	Bits															
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x14C0	ICUNIT0e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x14C4	ICUNIT1e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x14E0	ICCONFIG0	31:16	reserved (*)															
		15:0	reserved (*)												ICMODE0			
0x14E4	ICCONFIG1	31:16	reserved (*)															
		15:0	reserved (*)												ICMODE1			
0x1500	ICMUX0	31:16	reserved (*)															
		15:0	reserved (*)								ICMUX0 <7:0>							
0x1504	ICMUX1	31:16	reserved (*)															
		15:0	reserved (*)								ICMUX1 <7:0>							
0x1540	ICPULS0	31:16	ICPULS0 <31:16>															
		15:0	ICPULS0 <15:0>															
0x1544	ICPULS1	31:16	ICPULS1 <31:16>															
		15:0	ICPULS1 <15:0>															
0x1560	ICPERIOD0	31:16	ICPERIOD0 <31:16>															
		15:0	ICPERIOD0 <15:0>															
0x1504	ICPERIOD0	31:16	ICPERIOD1 <31:16>															
		15:0	ICPERIOD1 <15:0>															

(*) reserved area has to be assigned with 0

Register ICUNITxe:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U							
	reserved							
23:16	U							
	reserved							
15:8	U							
	reserved							
7:0	U							R/W
	reserved							en

Bit 31 - 1 reserved (value 0 is written)
Bit 0 **ICUNITxe<0>** (default = 0)
Start and stop the IC-Unit
0 = stopped (default)
1 = started (operating measuring)

Register ICCONFIGx:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U							
	reserved							
23:16	U							
	reserved							
15:8	U							
	reserved							
7:0	U				R/W			
	reserved				ICMODEx <3:0>			

Bit 31 - 4 reserved (value 0 is written)
Bit 3 - 0 **ICMODEx<3:0>** (default = 0)
Determines the mode the IC Unit is working with
0 = Mode 0 performs continuous measurement of pulse and period duration of periodic signals (default)
1 - 15 = reserved (assign with 0)

Register ICMUXx:

<table>
<tr><th>Bit Range</th><th>Bit 31/23/15/7</th><th>Bit 30/22/14/6</th><th>Bit 29/21/13/5</th><th>Bit 28/20/12/4</th><th>Bit 27/19/11/3</th><th>Bit 26/18/10/2</th><th>Bit 25/17/9/1</th><th>Bit 24/16/8/0</th></tr>
<tr><td rowspan="2">31:24</td><td colspan="8">U</td></tr>
<tr><td colspan="8">reserved</td></tr>
<tr><td rowspan="2">23:16</td><td colspan="8">U</td></tr>
<tr><td colspan="8">reserved</td></tr>
<tr><td rowspan="2">15:8</td><td colspan="8">U</td></tr>
<tr><td colspan="8">reserved</td></tr>
<tr><td rowspan="2">7:0</td><td colspan="8">R/W</td></tr>
<tr><td colspan="8">ICMUXx <7:0></td></tr>
</table>

Bit 31 - 8 reserved (value 0 is written)

Bit 7 - 0 **ICMUXx <7:0>** (default = 0)

The register value determines the card peripheral the IC unit is applied to

0 = Optocoupler input IN00 (default)
1 = Optocoupler input IN01
.
.
.
31 = Optocoupler input IN31
255 - 32 = reserved -> assign with 0

Register ICPULSx:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R							
	ICPULSx <31:24>							
23:16	R							
	ICPULSx <23:16>							
15:8	R							
	ICPULSx <15:8>							
7:0	R							
	ICPULSx <7:0>							

Bit 31 - 0 **ICPULSx<31:0>**

From this register read out the last measured pulse duration in µs

Register ICPERIODx:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R							
	ICPERIODx <31:24>							
23:16	R							
	ICPERIODx <23:16>							
15:8	R							
	ICPERIODx <15:8>							
7:0	R							
	ICPERIODx <7:0>							

Bit 31 - 0 **ICPERIODx<31:0>**

From this register read out the last measured period duration in µs

10. Output Compare Unit

The board WASCO-PCIe8132 brings the option to the user to link a PWM function to the outputs or to return discrete pulses via the Output Compare Units. In this case, square-wave signals with a period duration of 2 to 2^{32} µs and a pulse duration of 1 to 2^{32} µs can be generated.
Attention: although the OC unit enables a resolution in a microsecond range, due to the output optocoupler specifications only graduations in millisecond range make sense.

10.1 PWM

The Output Compare Units enable the user to apply a PWM to any optocoupler output.

10.1.1 Operating principle

In order to realize the PWM, the OC Unit is using a writable 32-bit timer with adjustable period duration in µs (OCPERIODx) and a two-level compare register (OCUNITORx) to set the pulse duration in µs. If the OC Unit is deactivated, there is a LOW at the output (optocoupler output blocks). If the OC Unit is started in PWM mode, the timer starts counting in µs-clock and the OC output remains LOW. When the timer reaches the value in the register OCPERIODx, it will run over to the next cycle and start counting again at 0. Also at an overflow, the pipeline register connected to the timer will take-over the pulse duration configured in the register OCUNITORx, as soon as the OC output is set to HIGH (provided the pulse duration is not 0µs). When the timer value (OCTIMERx) matches the value of the pipeline register connected to the timer, the output will be set to LOW until the next timer overflow.

The application of the two-level pulse duration register ensures the complete return of each period prior to transfer, if the pulse duration is changed during OC operating.
If you want to skip the first period after starting the OC unit, in which no pulse at the output is emitted, you can preload the timer accordingly with another value than 0 (-> reduction of the period).

10.1.2 Calculation of the register values

OCPERIODx = Period_duration_in_µs + 1 [µs]

OCUNITORx = Pulse_duration_in_µs [µs]

OCTIMER = clocks [µs]

10.1.3 Application example

1. Deactivate the OC unit by clearing the corresponding bit in the register OCUNITxe
2.Connect the OC unit to the required optocoupler output. For this, select the source in the optocoupler OPTOOUTMUXx register (see chapter optocoupler output multiplexer)
3. Preload the OC timer of the OC unit. Here usually the value 0x00000000 is written to the register OCTIMERx.
4. Define the period duration of the PWM. For this, write the period duration to the register OCUNITORx as follows:
OCPERIODx = Period duration - 1 [µs]
5. Define the pulse duration. For this, write the pulse duration to the register OCPULSx as follows:
OCPULSx = Pulse duration [µs]
6. Select the mode of the OC unit. For using the PWM, the value 0 has to be written to the register OCCONFIGx
7. Activate the OC unit by setting the corresponding bit in register OCUNITe.

10.2 Pulse output

In addition to the PWM, the OC unit makes it possible to output separate µs-accurate pulses at the optocoupler outputs.

10.2.1 Functionality

To output separate positive pulses (= optocoupler enabled) you firstly have to configure the registers. Then, each time the en-bit is cleared and then set in the register OCUNITxe, you can issue a pulse as shown in the figure below:

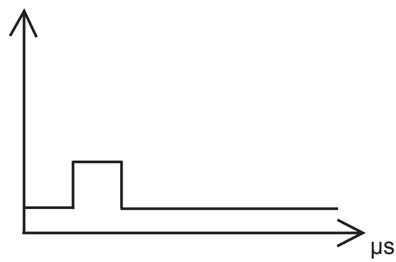


Fig. 10.2

To change the pulse duration, the OC unit always has to be deactivated (en-Bit in the OCUNITxe register cleared).

10.2.2 Computing of the register values

OCPERIODx = Pulse_duration_in_µs

10.2.3 Application example

1. How to configure the OC unit

a) Deactivate the unit by clearing (= 0) the en-Bit in the register OCUNITxe
b) Connect the OC unit to the required optocoupler output. For this, select the source in the OPTOOUTMUXx register (see chapter 6.4.2 optocoupler output multiplexer)
c) Preload the OC timer of the OC unit with the value 0. For this, write 0x00000000 to the register OCTIMERx.
d) Load the desired pulse duration to the register OCPERIODx (see also chapter 10.2.2 Computing of the register values)
e) Load the word 1 to the register OCUNITORx.
f) Select the mode Single Pulse by writing the value 1 to the mode section in the register OCCONFIGx.

2. How to output a pulse

a) Deactivate OC unit by clearing the enable bit in register OCUNITxe.
b) Activate OC unit by setting the enable bit in register OCUNITxe. As a result the pulse is applied to the selected output.

Offset-Address	Register Name	Bit Range	Bits															
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x15C0	OCUNIT0e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x15C4	OCUNIT1e	31:16	reserved (*)															
		15:0	reserved (*)															en
0x1600	OCTIMER0	31:16	OCTIMER0 <31:16>															
		15:0	OCTIMER0 <15:0>															
0x1604	OCTIMER1	31:16	OCTIMER1 <31:16>															
		15:0	OCTIMER1 <15:0>															
0x1620	OCUNITOR0	31:16	OCUNITOR0 <31:16>															
		15:0	OCUNITOR0 <15:0>															
0x1624	OCUNITOR1	31:16	OCUNITOR1 <31:16>															
		15:0	OCUNITOR1 <15:0>															
0x1660	OCPERIOD0	31:16	OCPERIOD0 <31:16>															
		15:0	OCPERIOD0 <15:7>															
0x1664	OCPERIOD1	31:16	OCPERIOD1 <31:16>															
		15:0	OCPERIOD1 <15:7>															
0x1680	OCCONFIG0	31:16	reserved (*)															
		15:0	reserved (*)												OCMODE0 <3:0>			
0x1684	OCCONFIG1	31:16	reserved (*)															
		15:0	reserved (*)												OCMODE1 <3:0>			

(*) reserved area has to be assigned with 0

Register OCUNITxe:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U							
	reserved							
23:16	U							
	reserved							
15:8	U							
	reserved							
7:0	U							R/W
	reserved							en

Bit 31 - 1 reserved (value 0 is written)
Bit 0 **OCUNITxe<0>** (default = 0)
Start or stop OC Unit
0 = stopped (default)
1 = started (performing measurements)

Register OCTIMERx:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W							
	OCTIMERx <31:24>							
23:16	R/W							
	OCTIMERx <23:16>							
15:8	R/W							
	OCTIMERx <15:8>							
7:0	R/W							
	OCTIMERx <7:0>							

Bit 31 - 0 **OCTIMERx<31:0>** (default = 0)
From this register read out or write to (e.g. for the initial state) the current value of the OC Timer x

Register OCUNITORx:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R							
	OCUNITORx <31:24>							
23:16	R							
	OCUNITORx <23:16>							
15:8	R							
	OCUNITORx <15:8>							
7:0	R							
	OCUNITORx <7:0>							

Bit 31 - 0 **OCUNITORx<31:0>** (default = 0)
Defines the pulse duration of the OC Unit x in µs
Pulse duration = OCUNITORx [µs]

Register OCPERIODx:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R							
	OCPERIODx <31:24>							
23:16	R							
	OCPERIODx <23:16>							
15:8	R							
	OCPERIODx <15:8>							
7:0	R							
	OCPERIODx <7:0>							

Bit 31 - 0 **OCPERIODx<31:0>** (default = 0)
Defines the period duration of the OC Unit x in µs
Period duration = OCPERIODx +1 [µs]

Register OCCONFIGx:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U							
	reserved							
23:16	U							
	reserved							
15:8	U							
	reserved							
7:0	U				R/W			
	reserved				OCMODEx <3:0>			

Bit 31 - 4 reserved (value 0 is written)

Bit 3 - 0 **OCMODEx<3:0>** (default = 0)

Determines the mode of the OC Unit

0 = Mode 0 -> Pulse Width Modulation (default)

1 = Mode 1 -> discrete pulse output

2 - 15 = reserved (assign 0)

11. Interrupt Controller

The interrupt controller is used to process the single interrupts from the various possible sources. It can enable single interrupt sources or can detect the sources of triggered interrupts.
The 32 bit register INTCON is the Central Unit as shown in fig 11.1. Here all possible interrupt sources (partially already processed) are merged.
If an interrupt is triggered, e.g. by an edge on an optocoupler input, this is passed to the first bit in the register INTCON. Whenever the register value of INTCON is nonzero (one or more interrupts are applied) this will be forwarded to INT. Thus, INT represents a type of gate register. The interrupt will be forwarded to the PC when the board‘s interrupt function is enabled (INTe = 1) and the register is reset. The interrupt line to the PC is blocked for any further interrupts when an interrupt has been triggered. To enable again the line, the source has to be determined and the trigger serviced. During this time it is possible to trigger further interrupts from other sources on the board (e.g. by other egde inputs or timers), but they will not be forwarded to the PC. When an interrupt trigger is serviced and the respective source is enabled again, the respective bit in register INTCON will be set to 0 automatically. All of the interrupt triggers being serviced and reset (INT = 0), the register INT can be cleared by setting the first bit in the INTr register and another interrupt can be forwarded to the PC.

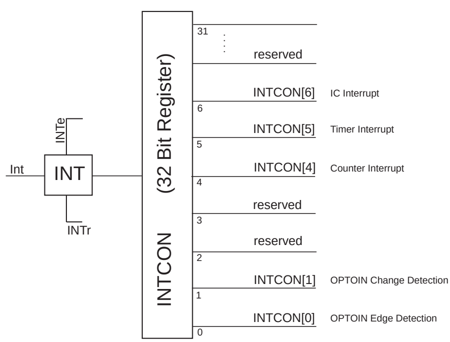


Fig. 11.1

As several sources of an optocoupler input edge interrupt are to consider, the source lines to the register INTCON usually are conditioned. This means, an additional 32 bit register can be applied to the respective bit of the register INTCON. In case of an edge detection of the optocoupler inputs, this is the register OPTOINF. In this register every bit represents one optocoupler input (see register description). Every single input can be armed individually to be an interrupt source (OPTOINFe) and enabled again after an interrupt is triggered and serviced (OPTOINFr).
The process completed, the respective bit in the register INTCON is set to 0 automatically.

Application

1. How to configure

a) Check whether or not all interrupt sources are cleared (INTCON = 0)
b) Enable single interrupt sources (see documentations of the respective peripherals)
c) Enable interrupt function (INTe = 1)

2. Interrupt routine

a) Identify interrupt source peripherals by reading INTCON and corresponding peripheral register if required
b) Clear interrupt
c) Check if there are any outstanding interrupts (INTCON = 0?)
d) if c) is the case, clear all other interrupts as well
e) Enable again the interrupt function (INTr = 1)

11.1 Port Addresses

Offset-Address	Register Name	Bit Range	Bits															
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x0280	INTe	31:16	reserved (*)															
		15:0	reserved (*)															en
0x0284	INTr	31:16	reserved (*)															
		15:0	reserved (*)															re
0x0288	INTCON	31:16	reserved (*)															
		15:0	reserved (*)								INT7	INT6	INT5	INT4	reserved (*)		INT1	INT0
0x028C	OPTONIFe	31:16	OPTOINIFe <31:16>															
		15:0	OPTOINIFe <15:0>															
0x0294	OPTOINIFr	31:16	OPTOINIFr <31:16>															
		15:0	OPTOINIFr <15:0>															
0x029C	OPTOINIF	31:16	OPTOINIF <31:16>															
		15:0	OPTOINIF <15:0>															
0x02A8	OPTOINICe	31:16	reserved (*)															
		15:0	reserved (*)															en
0x02AC	OPTOINICCe	31:16	OPTOINICCe<31:16>															
		15:0	OPTOINICCe<15:0>															
0x02B4	OPTOINICr	31:16	reserved (*)															
		15:0	reserved (*)															reset
0x02B8	OPTOINIC	31:16	reserved (*)															
		15:0	reserved (*)															OPTOINIC<0>
0x0340	COUNTIRe	31:16	COUNTIRe <31:16>															
		15:0	COUNTIRe <15:0>															
0x0344	COUNTIRr	31:16	COUNTIRr <31:16>															
		15:0	COUNTIRr <15:0>															

(*) reserved area has to be assigned with 0

<table>
<tr><th rowspan="2">Offset-Address</th><th rowspan="2">Register Name</th><th rowspan="2">Bit Range</th><th colspan="16">Bits</th></tr>
<tr><th>31/15</th><th>30/14</th><th>29/13</th><th>28/12</th><th>27/11</th><th>26/10</th><th>25/9</th><th>24/8</th><th>23/7</th><th>22/6</th><th>21/5</th><th>20/4</th><th>19/3</th><th>18/2</th><th>17/1</th><th>16/0</th></tr>
<tr><td rowspan="2">0x0348</td><td rowspan="2">COUNTIR</td><td>31:16</td><td colspan="16">COUNTIR<31:16></td></tr>
<tr><td>15:0</td><td colspan="16">COUNTIR<15:0></td></tr>
<tr><td rowspan="2">0x0360</td><td rowspan="2">TIMERIRe</td><td>31:16</td><td colspan="16">reserved (*)</td></tr>
<tr><td>15:0</td><td colspan="14">reserved (*)</td><td colspan="2">TIMERIRe<1:0></td></tr>
<tr><td rowspan="2">0x0364</td><td rowspan="2">TIMERIRr</td><td>31:16</td><td colspan="16">reserved (*)</td></tr>
<tr><td>15:0</td><td colspan="14">reserved (*)</td><td colspan="2">TIMERIRr<1:0></td></tr>
<tr><td rowspan="2">0x0368</td><td rowspan="2">TIMERIR</td><td>31:16</td><td colspan="16">reserved (*)</td></tr>
<tr><td>15:0</td><td colspan="14">reserved (*)</td><td colspan="2">TIMERIR<1:0></td></tr>
<tr><td rowspan="2">0x036C</td><td rowspan="2">ICUNITIRe</td><td>31:16</td><td colspan="16">reserved (*)</td></tr>
<tr><td>15:0</td><td colspan="14">reserved (*)</td><td colspan="2">ICUNITIRe<1:0></td></tr>
<tr><td rowspan="2">0x0370</td><td rowspan="2">ICUNITIRr</td><td>31:16</td><td colspan="16">reserved (*)</td></tr>
<tr><td>15:0</td><td colspan="14">reserved (*)</td><td colspan="2">ICUNITIRr<1:0></td></tr>
<tr><td rowspan="2">0x0374</td><td rowspan="2">ICUNITIR</td><td>31:16</td><td colspan="16">reserved (*)</td></tr>
<tr><td>15:0</td><td colspan="14">reserved (*)</td><td colspan="2">TIMERIR<1:0></td></tr>
</table>

(*) reserved area has to be assigned with 0

Register INTe:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U							
	reserved							
23:16	U							
	reserved							
15:8	U							
	reserved							
7:0	U							R/W
	reserved							en

Bit 31 - 1 reserved (write the vlaue 0)

Bit 0 **INTe<0>** (default = 0)
Enable or lock the card‘s interrupt function
0 = Interrupt locked (default)
1 = Interrupt enabled

Register INTr:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U							
	reserved							
23:16	U							
	reserved							
15:8	U							
	reserved							
7:0	U							W
	reserved							en

Bit 31 - 1 reserved (value 0 is written)

Bit 0 **INTr<0>**
The register INTCON is set to 0 by writing a 1 and a new interrupt can be triggered.

Register INTCON:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U							
	reserved							
23:16	U							
	reserved							
15:8	U							
	reserved							
7:0	U	R			U		R	
	reserved	INTCON <6:4>			reserved		INTCON <1:0>	

Bit 31 - 7 reserved (value 0 is written)
Bit 6 **INTCON<6>**: signals an interrupt from one of the OC Units
0 = no interrupt was triggered by an OC Unit
1 = one of the OC Units triggered an interrupt
Bit 5 **INTCON<5>**: signals an interrupt from one of the timers
0 = no interrupt was triggered by a timer
1 = one of the timers triggered an interrupt
Bit 4 **INTCON<4>**: signals an interrupt from one of the counters
0 = no interrupt was triggered by a counter
1 = one of the counters triggered an interrupt
Bit 3 - 2 reserved (value 0 is written)
Bit 1 **INTCON<1>**: signals an interrupt triggered by a change of an opto-coupler input applied to this interrupt
0 = no interrupt was triggered by a change of the optocoupler inputs
1 = interrupt has been triggered by a change of the optocoupler inputs
Bit 0 **INTCON<0>**: signals an interrupt triggered by a rising edge of an optocoupler input applied to this interrupt
0 = no interrupt was triggered by an edge of the optocoupler inputs
1 = interrupt was triggered by an edge of the optocoupler inputs

Register OPTOINIFe

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W							
	OPTOINIFe <31:24>							
23:16	R/W							
	OPTOINIFe <23:16>							
15:8	R/W							
	OPTOINIFe <15:8>							
7:0	R/W							
	OPTOINIFe <7:0>							

Bit 31 - 0 **OPTOINIFe**<31:0> In this register section single optocoupler inputs can be enabled as a source to trigger an interrupt on a positive edge. Every bit corresponds to an optocoupler input (e.g. IN00 => OPTOINIFe<0>, IN13 => OPTOINIFe<13>). If a bit is 1, the function for edge interrupt of the optocoupler input is enabled, if it is 0 the function is blocked.

Register OPTOINIFr

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	W							
	OPTOINIFr <31:24>							
23:16	W							
	OPTOINIFr <23:16>							
15:8	W							
	OPTOINIFr <15:8>							
7:0	W							
	OPTOINIFr <7:0>							

Bit 31 - 0 **OPTOINIFr<31:0>** Each bit corresponds to an optocoupler input. (e.g IN00 => OPTOINIFr<0>, IN13 => OPTOINIFr<13>). If an edge interrupt has been triggered on an optocoupler input, its signal bit in the OPTOINIF register has to be reset. This is done by setting (= 1) the corresponding OPTONIFr bit. The OPTONIFr bits are set to 0 automatically after reset.

Register OPTOINIF

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R							
	OPTOINIF <31:24>							
23:16	R							
	OPTOINIF <23:16>							
15:8	R							
	OPTOINIF <15:8>							
7:0	R							
	OPTOINIF <7:0>							

Bit 31 - 0 **OPTOINIF<31:0>** indicates, whether there is a rising edge on one of the optocoupler inputs. Every bit corresponds to an opto-coupler input (e.g. IN00 => OPTOINIF<0>, IN13 => OPTOINIF<13>). A 1 in the respective bit it indicates, that there has been a rising edge on the input since the last reset, a 0 stands for no edge applied.

Register OPTOINICe:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U							
	reserved							
23:16	U							
	reserved							
15:8	U							
	reserved							
7:0	U							R/W
	reserved							en

Bit 31 - 1 reserved (value 0 is written)

Bit 0 **OPTOINICe<0>** (default = 0) Enable interrupt functions to detect changes on the optocoupler inputs
0 = Interrupt locked (default)
1 = Interrupt enabled

Register OPTOINICCe:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W							
	OPTOINICCe<31:24>							
23:16	R/W							
	OPTOINICCe<23:16>							
15:8	R/W							
	OPTOINICCe<15:8>							
7:0	R/W							
	OPTOINICCe<7:0>							

Bit 31 - 0 **OPTOINICCe<31:0>** (default = 0)
Enable or lock single optocoupler inputs for interrupt function to detect changes on the optocoupler inputs. Each bit corresponds to an optocoupler input.
(e.g. IN00 => OPTOINICCe<0>, IN13 => OPTOINICCe<13>)
0 = Interrupt locked (default)
1 = Interrupt enabled

Register OPTOINICr:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U							
	reserved							
23:16	U							
	reserved							
15:8	U							
	reserved							
7:0	U							W
	reserved							re

Bit 31 - 15 reserved (value 0 is written)
Bit 0 **OPTOINICr<0>** (default = 0)
If an interrupt has been triggered by a change on the optocoupler inputs, the source register OPTOINIC has to be set to 0 again. This is done by setting (=1) the OPTOINICr bit. The OPTOINICr bit is set to 0 automatically after reset.

Register OPTOINIC:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U							
	-							
23:16	U							
	-							
15:8	U							
	-							
7:0	U							R
	-							OPTOINIC<0>

Bit 31 - 1 undefined

Bit 0 **OPTOINIC<0>** indicates a change on an enabled optocoupler input
0 = no change
1 = change to an enabled optocoupler input

Register COUNTIRe:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W							
	COUNTIRe<31:24>							
23:16	R/W							
	COUNTIRe<23:16>							
15:8	R/W							
	COUNTIRe<15:8>							
7:0	R/W							
	COUNTIRe<7:0>							

Bit 31 - 0 **COUNTIRe<31:0>** (default = 0)
This enables the interrupt functions of the counters. Each bit corresponds to a counter.
(e.g. Counter 0 => COUNTIRe<0>, Counter 13 => COUNTIRe<13>)
0 = Interrupt locked (default)
1 = Interrupt enabled

Register COUNTIRr:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W							
	COUNTIRr<31:24>							
23:16	R/W							
	COUNTIRr<23:16>							
15:8	R/W							
	COUNTIRr<15:8>							
7:0	R/W							
	COUNTIRr<7:0>							

Bit 31 - 0 **COUNTIRr<31:0>**
Each bit corresponds to a counter (e.g. Counter 0 => COUNTIRr<0>, Counter 13 => COUNTIRr<13>). If an interrupt has been triggered from a counter, its signal bit in the COUNTIR register has to be reset. This is done by setting (= 1) the corresponding COUNTIRr bit. The COUNTIRr bits are set to 0 automatically after reset.

Register COUNTIR:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U							
	COUNTIR<31:24>							
23:16	U							
	COUNTIR<23:16>							
15:8	R/W							
	COUNTIR<15:8>							
7:0	R/W							
	COUNTIR<7:0>							

Bit 31 - 0 **COUNTIR<31:0>** indicates whether an interrupt has been triggered from a counter. Each bit corresponds to a counter (e.g. Counter 0 => COUNTIR<0>, Counter 13 => COUNTIR<13>)
0 = no interrupt
1 = interrupt triggered

Register TIMERIRe:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U							
	reserved							
23:16	U							
	reserved							
15:8	U							
	reserved							
7:0	U						R/W	
	reserved						TIMERIRe <1:0>	

Bit 31 - 2 reserved (value 0 is written)

Bit 1 - 0 **TIMERIRe<1:0>** This enables the interrupt functions of the timers. Each bit corresponds to a timer (e.g. timer 0 => TIMERIRe<0>, timer 1 => TIMERIRe<1>)
0 = Interrupt locked (default)
1 = Interrupt enabled

Register TIMERIRr:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U							
	reserved							
23:16	U							
	reserved							
15:8	U							
	reserved							
7:0	U						R/W	
	reserved						TIMERIRr <1:0>	

Bit 31 - 2 reserved (value 0 is written)

Bit 1 - 0 **TIMERIRr<1:0>** Each bit corresponds to a timer (e.g. Timer 0 => TIMERIRr<0>, Timer 1 => TIMERIRr<1>). If an interrupt has been triggered from a timer, its signal bit in the TIMERIR register has to be reset. This is done by setting (= 1) the corresponding TIMERIRr bit. The TIMERIRr bits are set to 0 automatically after reset.

Register TIMERIR:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U							
	reserved							
23:16	U							
	reserved							
15:8	U							
	reserved							
7:0	R							
	reserved						TIMERIR <1:0>	

Bit 31 - 2 reserved (value 0 is written)

Bit 1 - 0 **TIMERIR<1:0>** indicates whether an interrupt has been triggered from a timer. Each bit corresponds to a timer
(e.g. Timer 0 => TIMERIR<0>, Timer 1 => TIMERIR<1>)
0 = no interrupt
1 = interrupt triggered

Register ICUNITIRe:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U							
	reserved							
23:16	U							
	reserved							
15:8	U							
	reserved							
7:0	U						R/W	
	reserved						ICUNITIRe<1:0>	

Bit 31 - 2 reserved (value 0 is written)

Bit 1 - 0 **ICUNITIRe<1:0>** This enables the interrupt functions of the IC Units. Each bit corresponds to an IC Unit
(e.g. IC Unit 0 => ICUNITIRe<0>, IC Unit 1 => ICUNITIRe<1>)
0 = Interrupt locked (default)
1 = Interrupt enabled

Register ICUNITIRr:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U							
	reserved							
23:16	U							
	reserved							
15:8	U							
	reserved							
7:0	U						R/W	
	reserved						ICUNITIRr<7:0>	

Bit 31 - 2 reserved (value 0 is written)

Bit 1 - 0 **ICUNITIRr<1:0>** Each bit corresponds to an IC Unit (e.g. IC Unit 0 => ICUNITIRr<0>, IC Unit 1 => ICUNITIRr<1>). If an interrupt has been triggered from a IC Unit, its signal bit in the ICUNITIR register has to be reset. This is done by setting (= 1) the corresponding ICUNITIRr bit. The ICUNITIRr bits are set to 0 automatically after reset.

Register ICUNITIR:

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U							
	reserved							
23:16	U							
	reserved							
15:8	U							
	reserved							
7:0	R							
	reserved						ICUNITIR <1:0>	

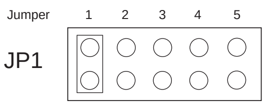
Bit 31 - 2 reserved (value 0 is written)

Bit 1 - 0 **ICUNITIR<1:0>** indicates whether an interrupt has been triggered from a IC Unit. Each bit corresponds to a IC Unit (e.g. IC Unit 0 => ICUNITIR<0>, IC Unit 1 => ICUNITIR<1>).

0 = no interrupt

1 = interrupt triggered

12. Board Identification



The board identification is used to differentiate between several PC cards of the same type on the computer. This is done by a jumper block, which can be read by software.

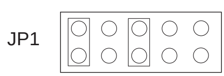
The board identification to be read consists of one word (16 Bit) and is structured as follows:

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Jumper												5	4	3	2	1
Board ID Register	0	0	0	0	0	0	0	0	0	0	0	x	x	x	x	x

„x“ is „1“, if the jumper is set, otherwise „0“

The jumper setting of the jumper block JP1 can be read out by means of the read command. The unused bits are basically „0“, a set jumper is read as „1“.

E.g.



(Jumper 1 und 3 set)

Result of the read command: $0005

12.1 Port Addresses

Offset-Address	Register Name	Bit Range	Bits															
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0xOFF8	BOARDID	31:16	reserved (*)															
		15:0	reserved (*)											Board ID				

(*) reserved area has to be assigned with 0

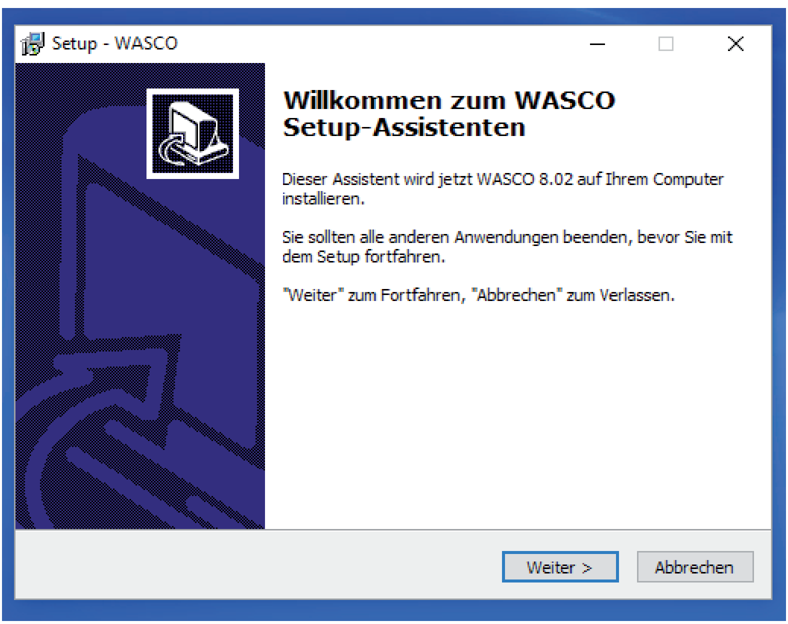
13. Programming under Windows®

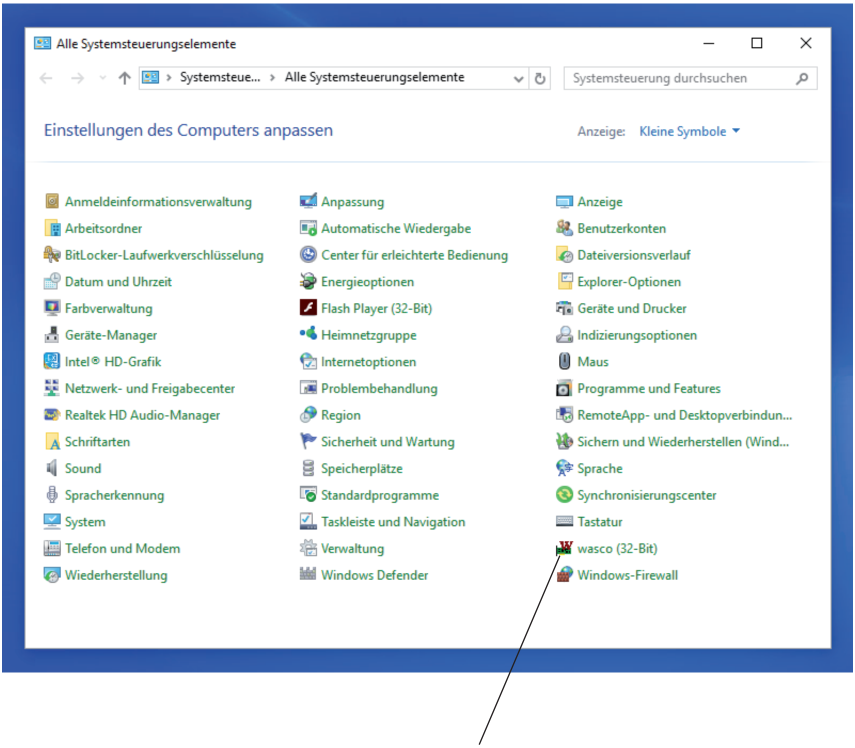
13.1 Installation of the Windows® driver

In order to implement the card under Windows®, it is necessary to install a special driver, which allows access to the card. The operating system under Windows® 10, 8 and 7 automatically reports after starting the PC, that a new hardware component has been found. In this case, insert the data medium and instruct to the system to install the driver files herefrom. If the operating system does not respond, the driver also can be installed in the Device Manager.

13.2 Installation of the Windows® development files

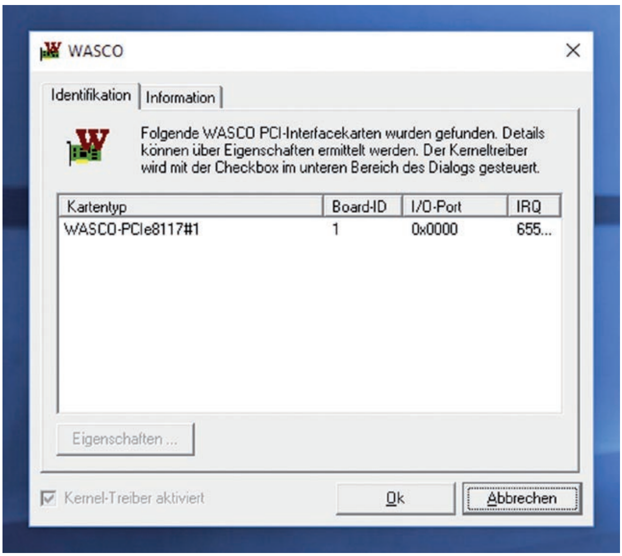
For installation of the development files, please run the file "Setup.exe" in the folder driver on the accompanying CD and follow the installation instructions.





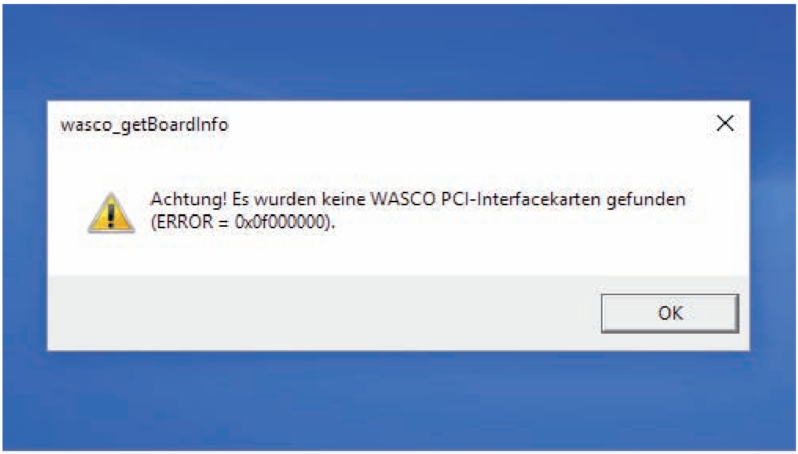
Once the driver and development files have been installed completely, you will find an icon in the control panel of your computer to localize all **wasco**® PCI and PCIe cards available in the system.

Start the card query by double-clicking the "**wasco**®"- Icon. Following window appears: (An OPTOIO-PCIe16ULTRA is used as an example)



If your card has been detected in the system, the board name WASCO-PCIe8132, Board-ID, I/O address as well as the possible interrupt number of the respective card are displayed in this window. Furthermore, the driver version and the location of the driver file can be queried via the „Information“ tab.

If your card was not detected, following error message will be displayed:



Please find more about the possible causes in the chapter Troubleshooting.

13.3 Programming the OPTOIO-PCIe32 with wasco® driver

After installation of the development files of Kithara by means of the setup program the folder **wasco** contains of the relevant development files and the sample programs. Further sample programs specified for access to the OPTOIO-PCIe32 you can find on the enclosed CD or please visit our homepage.

Programming the hardware components of the OPTOIO-PCIe32 is realized by access to Memory Mapped I/O addresses which depend on the base address assigned by the system's BIOS for the OPTOIO-PCIe32. Find a more detailed description for programming in the driver documentation.

13.4 Access to the OPTOIO-PCIe32ULTRA

The acess to the OPTOIO-PCIe32ULTRA is done exclusively via board name (card type) WASCO-PCIe8132

13.5 Assignment of the Memory Mapped I/O Addresses

The Memory Mapped I/O addresses of the single hardware components depend on the base address, as shown in following table using a few examples:

Port/Register	BA + Offset	RD/WR
Optocoupler Input Port (IN00...IN31)	BA + $0	RD
Optocoupler Output Port (OUT00...OUT31)	BA + $8	RD/WR
Board Identification	BA+ $FF8	RD

Attention! The driver‘s offset constants only work for PCI cards.

13.6 Compatibility to OPTOIO-PCI32STANDARD

Developping the OPTOIO-PCIe32ULTRA and its drivers special regard was attended to use identical accesses as to OPTOIO-PCI32. This enables you to swapp from PCI to PCIe with existing programs in a very easy way. The driver (as from version 8.02) is usable for PCI as well as for PCIe.

What has been changed and what has to be changed respectively for the PCIe board:

1. The board‘s name switched from „OPTOIO-PCI32“ to „OPTOIO-PCIe32ULTRA“
2. The functions to access to port addresses for the PCIe board are given as „wasco_outputPCIeW“ and „wasco_inputPCIeW“
3. The offsets to access to the Memory Mapped I/O addresses changed. (Constants are usable for PCI only).
4. The setting of jumper block JP1 can be monitored via an additional address. The jumper can be used for example for the identification of the OPTOIO-PCIe32ULTRA in case your computer registers more than one board.

14. Linux® Programming

To use the board under Linux®, you can find a Linux wasco® driver on the supplied CD or on our website. This is in code form and therefore can be changed and adapted by the customer at any time.
The driver does not include interrupt handling, but this can be added by the customer at any time. The reason for this is that the individual interrupt handling should be processed in the Kernel module.

14.1 Installing the Linux® driver

To apply the card under Linux® a special driver has to be installed, that enables access to the card. Insert the data medium and copy the folder of the Linux driver to your system. For installation, follow the instructions of the readme file.

14.2 Supported Linux-Distributions/Kernelversions

The wasco® driver has been tested in the following environments:

Ubuntu® 18.04.4 LTS (Kernel: 5.3.0)

14.3 Programming the OPTOIO-PCIe32 with wasco® driver

Programming the hardware components of the OPTOIO-PCIe32 is realised by accessing Memory Mapped I/O addresses which depend on the base address assigned by the system‘s BIOS for the OPTOIO-PCIe32.
The access is done via the functions pread und pwrite. For this, under programming language C and C/C++ no further external libraries are required. Examples for the exact access to the OPTOIO-PCIe32 can be found on the enclosed CD as well as on our homepage.
The Linux wasco® driver does not include interrupt handling, but this can be added by the customer at any time.

14.4 Access to the OPTOIO-PCIe32ULTRA

The access to the OPTOIO-PCIe32ULTRA is done exclusively via the board name (type of card) WASCO-PCIe8132

14.5 Assignment of the Memory Mapped I/O addresses

The Memory Mapped I/O addresses of the single hardware components depend on the base address according to following table:

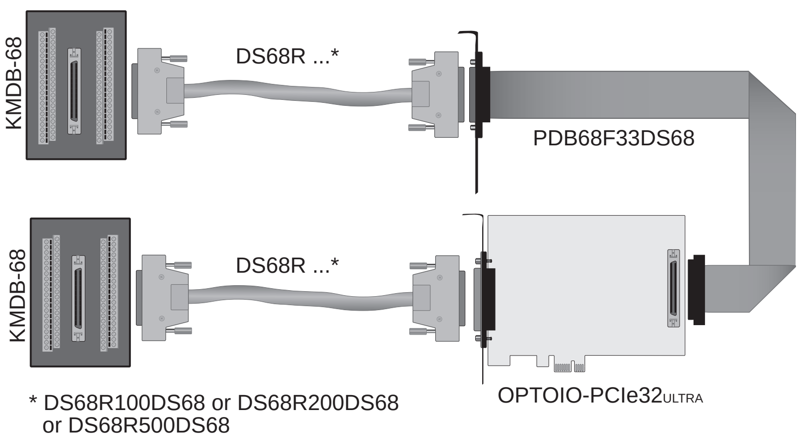
Port/Register	BA + Offset	RD/WR
Optocoupler input port (IN00...IN31)	BA + $0	RD
Optocoupler output port (OUT00...OUT31)	BA + $8	RD/WR
Board Identification	BA+ $FF8	RD

15. Accessories

15.1 Matching wasco® accessories

Connecting parts	EDP-No.
PDB68F33DS68 Ribbon cable	A-498600
KMDB-68 Terminal Module	A-494800
DS68R100DS68 Connecting wire (1 meter)	A-492200
DS68R200DS68 Connecting wire (2 meter)	A-492400
DS68R500DS68 Connecting wire (5 meter)	A-492800

15.2 Connection Technique (application examples)



15.3 Single components for customized assembly

Connection parts	EDP-No.
SCSI-II plug 68-pin for flat ribbon cable	A-553200
SCSI-II socket 68-pin for flat ribbon cable	A-557200
Slot bracket with cutout for 68-pin connector male/female	A-577800
Flat ribbon cable 68-pin	A-572800

16. Troubleshooting

Following you can find a brief compilation of the most common known causes of errors that may occur during starting-up or while running the OPTOIO-PCIe32.
Please check these points before you contact your dealer or distributor to solve your problem:

1st Is OPTOIO-PCIe32properly inserted to the connector ?

2nd Are all cable connections all right?

3rd Did your system detect the card correctly?
Please check all settings of your computer or contact your system administrator. (As this are BIOS settings of the computer we cannot expand on this issue. We refer to your systems manual.)

4th Did you install the latest driver version for the wasco® drivers?
Updates you can find here: http://www.messcomp.com

17. Specifications

Optocoupler Inputs
Optocouplers
32 channels, optically isolated
Galvanic isolation even between every single channel with each two separate connectors
Overvoltage protection by protection diodes
Two different input voltage ranges selectable by jumpers:

Range 1	high = 14..30 Volt
	low = 0..2 Volt
Range 2:	high = 5..15 Volt
	low = 0..1 Volt

Input frequency: ma. 10 kHz

Optocoupler Outputs
Optocouplers
32 channels, optically isolated, socketed
Galvanic isolation even between every single channel with each two separate connectors
Overvoltage protection by protection diodes
Output current max. 150mA
Output frequency ca 1 KHz
Voltage collector-emitter: max. 50V
Voltage emitter-collector: max. 0,1V

Board Identification
Jumper block with five pairs of contact pins

Connection plug
2 * 68-pin SCSI connectors

Bus system
32-Bit PCIe Bus (32 Bit data access)

Measurements of the Board
157 mm x 111 mm (l x b)
standard height, half length card
Multilayer PCB

Other
Control LEDs for power supply

18. Product Liability Act

Information about Product Liability

The Product Liability Act (Act on Liability for Defective Products - ProdHaftG) in Germany regulates the manufacturer‘s liability for damages caused by defective products.

The obligation to pay compensation can be given, if the product’s presentation could cause a misconception of safety to a non-commercial end-user and also if the end-user is expected not to observe the necessary safety instructions handling this product.

It must therefore always be verifiable, that the non-commercial end-user was made familiar with the safety rules.

In the interest of safety, please always advise your non-commercial customer of the following safety instructions:

Safety instructions

The valid VDE-instructions must be observed, when handling products that come in contact with electrical voltage.

Especially the following instructions must be observed:
VDE100; VDE0550/0551; VDE0700; VDE0711; VDE0860.

The instructions are available from:
Vde-Verlag GmbH
Bismarckstr. 33
10625 Berlin

* unplug the power plug before you open the unit or make sure, there is no current to/in the unit.

* You only may put into operation any components, boards or devices if they have been installed inside a secure touch-protected casing before. During installation there must be no current to the equipment.

* Make sure that the device is disconnected from the power supply before using any tools on any components, boards or equipment. Any electric charges stored in components in the device are to be discharged prior.

* Live cables or wires, which are connected with the unit, the components or the boards, must be tested for insulation defects or breaks. In case of any defect the device must be immediately taken out of operation until the defective cables have been replaced.

* When using components or boards you must strictly comply with the characteristic data for electrical sizes shown in the related description

* As a non-commercial end-user, if it is not clear whether or not the electrical characteristic data given in the provided description apply to a component you must consult an expert.

Furthermore, the compliance with building and safety instructions of all kinds (VDE, TÜV, industrial injuries corporation, etc.) is subject to the user/customer.

19. CE Confirmation

This is to certify, that the product

OPTOIO-PCIe32$_{ULTRA}$
EDP Number A-840810

comply with the requirements of the relevant EMC directives 2014/30/EU. This declaration will lose its validity, if the instructions given in this manual for the intended use of the products are not fully complied with.

Following standards are regarded:

EN 55011: 2009 + A1. 2010 (Group 1, Class A)
EN 55022: 2010 / AC: 2011
EN 55024: 2010
EN 61000-6-4: 2007 + A1: 2011
EN 61000-6-2: 2005 / AC: 2005
(EN 6100-4-2: 2008; EN 6100-4-3: 2006 + A1: 2007 + A2; EN 6100-4-4: 2012; EN 6100-4-5: 2014; EN 6100-4-6: 2013; EN 6100-4-8: 2009; EN 6100-4-11: 2004)

The following manufacturer is responsible for this declaration:

Messcomp Datentechnik GmbH
Neudecker Str. 11
83512 Wasserburg

given by

Dipl.Ing.(FH) Hans Schnellhammer

Wasserburg, 19.03.2019

Reference system for intended use

This PC expansion card is not a stand-alone device. The CE-conformity only can be assessed when additional computer components are in use simultaneously. Thus the CE conformity only can be confirmed when using the following reference system for the intended use of the PC expansion card:

Control Cabinet::	Vero IMRAK 3400	804-530061C 802-563424J 802-561589J
19" Casing:	Vero PC Casing	145-010108L
19" Casing:	additional Electronic	519-112111C
Motherboard:	ASUS P5G41-M LE	
Interface:	OPTOIO-PCIe32ULTRA	A-840810