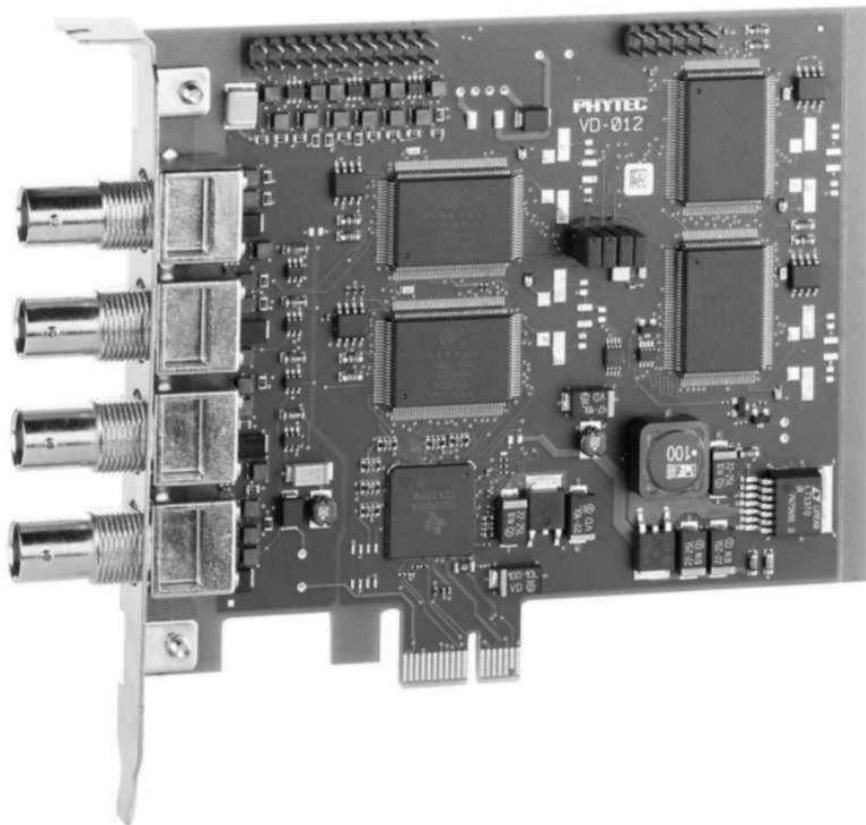


# ***PCI* Grabber-4x4**



## **Hardware-Manual**

**Edition January 2009**

A product of a PHYTEC Technology Holding company

In this manual are descriptions for copyrighted products that are not explicitly indicated as such. The absence of the trademark (™) and copyright (©) symbols does not imply that a product is not protected. Additionally, registered patents and trademarks are similarly not expressly indicated in this manual.

The information in this document has been carefully checked and is believed to be entirely reliable. However, PHYTEC Messtechnik GmbH assumes no responsibility for any inaccuracies. PHYTEC Messtechnik GmbH neither gives any guarantee nor accepts any liability whatsoever for consequential damages resulting from the use of this manual or its associated product. PHYTEC Messtechnik GmbH reserves the right to alter the information contained herein without prior notification and accepts no responsibility for any damages which might result.

Additionally, PHYTEC Messtechnik GmbH offers no guarantee nor accepts any liability for damages arising from the improper usage or improper installation of the hardware or software. PHYTEC Messtechnik GmbH further reserves the right to alter the layout and/or design of the hardware without prior notification and accepts no liability for doing so.

© Copyright 2009 PHYTEC Messtechnik GmbH, D-55129 Mainz.

Rights - including those of translation, reprint, broadcast, photomechanical or similar reproduction and storage or processing in computer systems, in whole or in part - are reserved. No reproduction may occur without the express written consent from PHYTEC Messtechnik GmbH.

	EUROPE	NORTH AMERICA
Address:	PHYTEC Technologie Holding AG Robert-Koch-Str. 39 D-55129 Mainz GERMANY	PHYTEC America LLC 203 Parfitt Way SW, Suite G100 Bainbridge Island, WA 98110 USA
Ordering Information:	+49 (800) 0749832 <a href="mailto:order@phytec.de">order@phytec.de</a>	1 (800) 278-9913 <a href="mailto:sales@phytec.com">sales@phytec.com</a>
Technical Support:	+49 (6131) 9221-31 <a href="mailto:support@phytec.de">support@phytec.de</a>	1 (800) 278-9913 <a href="mailto:support@phytec.com">support@phytec.com</a>
Fax:	+49 (6131) 9221-33	1 (206) 780-9135
Web Site:	<a href="http://www.phytec.de">http://www.phytec.de</a>	<a href="http://www.phytec.com">http://www.phytec.com</a>

1<sup>st</sup> Edition January 2009

---

<b>1</b>	<b>Introduction .....</b>	<b>1</b>
<b>2</b>	<b>Delivery Contents/ Technical Data .....</b>	<b>4</b>
2.1	Accessories .....	5
2.2	Technical Data VD-012(-X1)(-X2) .....	6
2.3	Field of Applications and Safety Regulations .....	10
2.4	Addresses and Resources .....	11
2.5	Socket Pinout .....	13
2.5.1	Composite Inputs .....	15
2.5.2	S-Video Connection .....	16
2.6	I <sup>2</sup> C Interface .....	17
2.7	Jumper on pin header row X900 .....	17
2.8	Notes on CE-Conformance and Immunity against Interference .....	18
2.9	Option Port .....	19
<b>3</b>	<b>Installation of the Grabber Card .....</b>	<b>20</b>
3.1	Installing the Grabber Card .....	20
3.2	Installing the Driver .....	22
3.2.1	Additional Drivers (optional) .....	24
3.3	Installing the Demo Program .....	25
<b>4</b>	<b>Connecting Video Sources .....</b>	<b>26</b>
4.1	Possible Video Connections .....	28
4.1.1	The S-Video Cable .....	30
4.1.2	The Composite Cable .....	30
4.2	Extension Card VZ-012 .....	31
4.3	Overview about all video inputs .....	34
<b>5</b>	<b>Start-Up of the Grabber with Demo Programs .....</b>	<b>37</b>
5.1	The parallel image processing .....	45
5.2	Demo Program Description .....	48
5.3	Image Control .....	54
5.4	Additional Functions Under <i>Image</i> .....	55
5.5	Crosshair function (Overlay) .....	56
5.6	Special Functions .....	56
5.7	Storing Images, Ending the Program .....	64
5.8	Getting Started with Linux .....	64
<b>6</b>	<b>Driver Software .....</b>	<b>66</b>
6.1	Technical Basics .....	67
6.1.1	Block Diagram of the pciGrabber-4x4 .....	67
6.1.2	The Videosignal and Digitization .....	70
6.1.3	Transfer and storage of color .....	72
6.1.4	Data storage by DMA and RISC-Program .....	74
6.2	Driver for Microsoft Windows .....	78
6.2.1	Requirements .....	79

---

6.2.2	Application of the Device Driver for Windows NT4.0	80
6.2.3	Application of the Device Driver for Windows 2000 / XP / VISTA.....	84
6.2.4	Application of the DLL.....	85
6.2.5	Application of the Windows XP/VISTA™ Windows NT4.0™ / Windows 2000™ DLLs.....	86
6.2.6	Programming under Delphi .....	87
6.2.7	Description of the DLL in Existing Functions.....	89
<b>7</b>	<b>Trouble-Shooting .....</b>	<b>143</b>
<b>Index</b>	<b>.....</b>	<b>148</b>

## Index of Figures

Figure 1:	Accessory Cables .....	5
Figure 2:	Connectors of the pciGrabber-4x4 (VD-012) .....	13
Figure 3:	Connectors of the pciGrabber-4x4 (VD-012-X1) .....	14
Figure 4:	Connectors of the pciGrabber-4x4 (VD-012-X2) .....	14
Figure 5:	Pin Formation of the Option Port .....	19
Figure 6:	Inserting the Card into the PCI Express Slot .....	21
Figure 7:	PHYTEC Install Menu .....	25
Figure 8:	Overview of the pciGrabber-4x4 Connectors (VD-012) .....	26
Figure 9:	Overview of the pciGrabber-4x4 Connectors (VD-012-X1) ...	27
Figure 10:	Overview of the pciGrabber-4x4 Connectors (VD-012-X2) ...	27
Figure 11:	Video Connector Cables - (Description and PHYTEC Order Number) .....	28
Figure 12:	Connectors for the pciGrabber-4x4 .....	29
Figure 13:	Extension Card VZ-012 with ribbon cable .....	31
Figure 14:	How to connect the extension cards .....	32
Figure 15:	Jumper settings for three VZ-012 for VD-012 und VD-012-X133	
Figure 16:	Jumper settings for VZ-012 for VD-012-X2 .....	33
Figure 17:	Video inputs VD-012 .....	34
Figure 18:	Video inputs VD-012-X1 .....	35
Figure 19:	Video inputs VD-012-X2 .....	35
Figure 20:	Overview of the Demo Program .....	37
Figure 21:	Basic Settings Menu .....	38
Figure 22:	Menu Option: Image .....	40
Figure 23:	Configuring the Image Parameters .....	41
Figure 24:	Live Image from the Video Source .....	43
Figure 25:	Overloaded PCI bus .....	47
Figure 26:	„Image Setting“ Menu .....	48
Figure 27:	Creating a Full Image: Two Fields, Each with 7 rows .....	51

Figure 28: Comb Effect That Occurs with Quick Moving Objects .....	52
Figure 29: The Image Control Window .....	54
Figure 30: Histogram.....	56
Figure 31: Color Meter .....	58
Figure 32: Arithmetics Menu .....	60
Figure 33: Selecting the Normalization Factor .....	61
Figure 34: Number of Images .....	61
Figure 35: Option Port Menü .....	62
Figure 36: Option Port Menü (Jumper).....	63
Figure 37: Block diagram VD-012.....	67
Figure 38: Block diagram VD-012( part 2).....	68
Figure 39: InterlacedImage (Example with 9 Lines) .....	70
Figure 40: Fields and Frames .....	71
Figure 41: Moving Objects Cause Comb Effects.....	71
Figure 42: Pixel- and Control Data Flow (Overview).....	76
Figure 43: Directory for Window's Driver .....	78
Figure 44: Windows NT Registration Editor .....	81
Figure 45: Entering a Device Driver .....	81
Figure 46: Configuring the Driver.....	82
Figure 47: Scaling and Cropping.....	112
Figure 48: Example of Scaling: Only the ppl Value is Different.....	113
Figure 49: Color Format of the pciGrabber-4x4 .....	120
Figure 50: Return Values of ,Data_Present' .....	126
Figure 51: Timing Diagram of the Return Parameter of ,Data_Present()126	

## Index of Tables

Table 1:	Pin Assignments of the Model VD-012 .....	15
Table 2:	Pin Assignments of the Model VD-012-X1.....	16
Table 3:	Pin Assignments of the Model VD-012-X2.....	16
Table 4	Connecting the I <sup>2</sup> C Interface to the Combi Socket .....	17
Table 5:	Pin Assignment for the Option Port X300 .....	19
Table 6:	Numbers of possible extension cards.....	32
Table 7	Example data volumes PAL 25fps.....	46
Table 8	Example data volumes NTSC 30fps .....	46
Table 9:	Required Memory Space of One Pixel for the Different Modi121	





## **1 Introduction**

Thanks for buying a pciGrabber-4x4 of the PHYTEC Messtechnik GmbH. This manual explains on the one hand how to install the PC-Card and on the other hand some information to the driver-software.

At the moment exists some kinds of pciGrabber4x4 models. To these belong three kinds, that are described in this manual. The first one is the model VD-012, the second VD-012-X1 and the third one is VD-012-X2. In the following overview is a summery of the two types and their models that are described in this manual.

<b>TYPE</b>	<b>Article-No.:</b>	<b>No. of Decoders</b>	<b>Bus-System</b>
pciGrabber-4x4	VD-012	4	x1 PCI Express
pciGrabber-4x4	VD-012-X1	4	x1 PCI Express
pciGrabber-4x4	VD-012-X2	2	x1 PCI Express



# **Part 1**

## **Installation and Start-Up**

## **2 Delivery Contents/ Technical Data**

The pciGrabber-4x4 includes the following upon delivery:

- A PCI Express-card
- Installation CD with
  - Demo software (Windows XP, NT4.0, 2000 and Windows VISTA)
  - Driver software for Windows XP, NT4.0, 2000 and Windows VISTA
  - Labview driver for photo processing applications using Labview (National Instruments, IMAQ – packet is required)
- the pciGrabber-4x4 manual

## 2.1 Accessories

The following pciGrabber-4x4 accessories may be ordered from PHYTEC:

- S-Video connector cable for connection of a color camera with a 4-pin Mini-DIN plug (S-Video output). Length, approx. 2 m – Order number: WK051
- BNC connector cable for connection of a camera with a BNC-plug. Order number: WK058 (2m) or WK039 (10m)
- Replacement fuse 1.6 AT TR5 for camera power supply (receptacle F2) – Order number KF012
- Replacement fuse 500 mAT TR5 for camera power supply (receptacle F1) – Order number KF014



Figure 1: Accessory Cables

## 2.2 Technical Data VD-012(-X1)(-X2)

### Physical

**Dimensions:** 125 x 90 x 20 mm plus face plate and slot  
*120 x 90 x 20 mm without panel edge*

**Data Bus:** x1 PCI Express bus  
(PCI Express Base Spec. Rev. 1.0a compliant)

**Power Supply:** +3.3V V (250 mA idle, 300 mA digitizing)  
(taken from the PCI Express bus)

### Numbers of Decoders:

Model VD-012: four Decoder

Model VD-012-X1 four Decoder

Model VD-012-X2 two Decoder

### Inputs:

#### Model VD-012:

4 composite video inputs, 75  $\Omega$ , 1 V<sub>ss</sub><sup>1</sup>  
*optional 12 composite video inputs 75  $\Omega$ , 1 V<sub>ss</sub>*

#### Model VD-012-X1:

4 S-Video input 75  $\Omega$  (0.7 V<sub>ss</sub> / 0.3 V<sub>ss</sub>)  
*optional 12 composite video inputs 75  $\Omega$ , 1 V<sub>ss</sub>*

#### Model VD-012-X2:

2 composite video inputs, 75  $\Omega$ , 1 V<sub>ss</sub><sup>1</sup>)  
2 S-Video input 75  $\Omega$  (0.7 V<sub>ss</sub> / 0.3 V<sub>ss</sub>)  
*optional 4 composite video inputs 75  $\Omega$ , 1 V<sub>ss</sub>*

**Video Format:** PAL (B,G,H,I), HTSC (M)  
or corresponding CCIR monochrome format

---

<sup>1</sup> : If an S-Video input is not being used, then an extra composite input is available for the user

---

<b>Synchronization:</b>	Composite sync. or sync to Y-signal external synchronization is not possible
<b>Data Format:</b>	16 Mio. colors RGB32, RGB24, YcrCb 4:2:2, YcrCb 4:1:1 64,000 colors RGB16 32,000 colors RGB15 256 gray shades Y8 gray scale
<b>Image Resolution:</b>	maximum 720 x 576 pixels (PAL) or 640 x 480 pixels (NTSC) Resolution is freely scalable in X and Y directions up to 14:1
<b>Image Transfer Rate:</b>	Half frame 20 ms (Odd or even field) Full frame 40 ms (Odd or even field) Image transfer to the main memory in real time (Bus master transfer)
<b>Used Resources:</b>	4 kByte main memory (register field) <i>per decoder</i> 4 kByte main memory (register field) <i>PCI Express-to-PCI-Bridge</i>
<b>Image control:</b>	Gamma correction (selectable) Brightness (+/- 50 %) Contrast (0 % ... 235 %) Color saturation (U: 0...201 %, V: 0...283 %) Hue (+/- 90°, only with NTSC)
<b>Image Storage:</b>	630 Byte FIFO on-board, Real time storage in the PC main memory Even-/odd field memory separated or Common full frame memory (selectable)

**Ports:** 8-bit parallel I/O, TTL signal (multi-purpose)

Parameter	Symbol	Min	Max
Input High Voltage	$V_{IH}$	2,0 V	5 V
Input Low Voltage	$V_{IL}$	-0,5 V	0,8 V
Output High Voltage	$V_{OH}$	2,4 V	-
Output Low Voltage	$V_{OL}$	-	0,4 V
Input Low Current	$I_{IL}$	-	-70 $\mu$ A
Input High Current	$I_{IH}$	-	70 $\mu$ A

1 I<sup>2</sup>C interface (Master)

Parameter	Symbol	Min	Max
Transmission rate <sup>1</sup>	$f_{I2C}$	99,2 kHz	396,8 kHz
Input High Voltage	$V_{IH}$	3,5 V	5 V
Input Low Voltage	$V_{IL}$	-0,5 V	1,5 V
Hysteresis	$V_{hys}$	0,2 V	
Input High Current	$I_{IH}$	-	10 $\mu$ A
Input Low Current	$I_{IL}$	-	-10 $\mu$ A
Output Low Voltage	$V_{OL}$	-	0,4 V

---

<sup>1</sup> : Both of the frequencies can be de-lactivated with software

---



**Connectors:**

Model VD-012

4 x BNC socket: composite video input

Pin header row 2x12: 12 composite inputs  
(*not on the face plate*)

Pin header row 2x6: GPIO port, 8 x TTL I/O  
(*not on the face plate*) I<sup>2</sup>C interface

Pin header row 2 x 4: 4 Jumper  
(*not on the face plate*)

Model VD-012-X1

4 x Mini-DIN socket: S-Video input

Pin header row 2x12: 12 composite inputs  
(*not on the face plate*)

Pin header row 2x6: GPIO port, 8 x TTL I/O  
(*not on the face plate*) I<sup>2</sup>C interface

Pin header row 2 x 4: 4 Jumper  
(*not on the face plate*)

Model VD-012

2 x BNC socket: composite video input

2 x Mini-DIN socket: S-Video input

Pin header row 2x12: 12 composite inputs  
(*not on the face plate*)

Pin header row 2x6: GPIO port, 8 x TTL I/O  
(*not on the face plate*) I<sup>2</sup>C interface

Pin header row 2 x 4: 4 Jumper  
(*not on the face plate*)

## **2.3 Field of Applications and Safety Regulations**

Please pay attention to the specified operation directives of the pciGrabber-4x4. Before starting operation please read carefully the manual.

- The pciGrabber-4x4 is designed for the digitization of video signals from standard TV-cameras. Signals from composite-video cameras can be processed, which comply with CCIR B, G, H, I and the sub standard CCIR B, G, H, I/PAL. In addition signals compliant to CCIR M/NTSC can be applied. Also separate luma and chroma signals from cameras, which correspond to the S-video standard are applicable.
- The digitization is achieved in real time. The image data are transferred via the PCI-bus of the PC. The transfer rate corresponds to the access time specified for the PCI master slot.
- The effective transfer rate must be sufficient to handle the volume of the image data, otherwise information might be lost.
- The pciGrabber-4x4 is determined for the utilization with a standard PC, which might be an office computer with an usual housing. The Grabber must have a reliable connection with the housing and the ground (PE).
- The board is designed to operate in dry and dustless environment. For applications in industrial environment you have to consider to take additional protective arrangements especially against radio interference and safety hazards.
- The application of the Grabber board in safety areas, for aviation and space and for nuclear or military purposes requires our examinations and our agreement.
- For industrial applications all rules for prevention of accidents and the rules of the employer's liability insurance association for electrical facilities are to observe.

- Before starting the operation of the Grabber board, it must be ensured, that the device is appropriate for the application and the specific location. In case of doubt, you should ask experts or the manufacturer.
- The product has to be protected from hard shocks and vibrations. Eventually the device has to be padded or cushioned, but the ventilation may not be obstructed.
- In need of repair only a specialist should be asked, who uses the original spare parts. For the installation of the Grabber, use only tested and approved cables. Only radio shielded cables should be utilized.

## **2.4 Addresses and Resources**

The pciGrabber-4x4 occupies a region of 4 kBytes in the main memory of the PC for the local registers per decoder. The addressing region is automatically specified by the BIOS and no hardware wiring (jumper setting) is required.

Several pciGrabber-4x4 can be installed in one system. The boards are configured automatically by the BIOS for different addresses.

It is not possible to determine which board is configured to which address. The base address of each board can be obtained by the PCI-BIOS. For the pciGrabber-4x4 the driver software determines the address via the BIOS and defines a device number. The driver also can determine the number of boards within the system and is able to control each board by its particular device number.

It is not possible to determine which board will be specified by which device number. This will be done only by the PCI-BIOS and the architecture of the PC-motherboard. Usually the addresses are allocated in sequence of the numbering of the PCI-slots. This might deviate for different manufacturers. To solve that problem you can use the Pin header row X900. With them you can set Jumper to give every Grabber an explicit address.

The pciGrabber-4x4 will activate an interrupt in case of certain events or a distinct operational status.

The Grabber is only a *single function device* so only the interrupt line /INTA of the PCI-bus can be used. To this PCI-bus-interrupt an interrupt of the PC is allocated via the BIOS, so that the program can react to this event.

The source of the interrupt can be determined from the interrupt status register of the Grabber.

Several boards can trigger the same interrupt /INTA, it must be determined which board caused the interrupt.

## 2.5 Socket Pinout

### Note:

The following description of the Grabber's connectors is intended only as a technical reference.

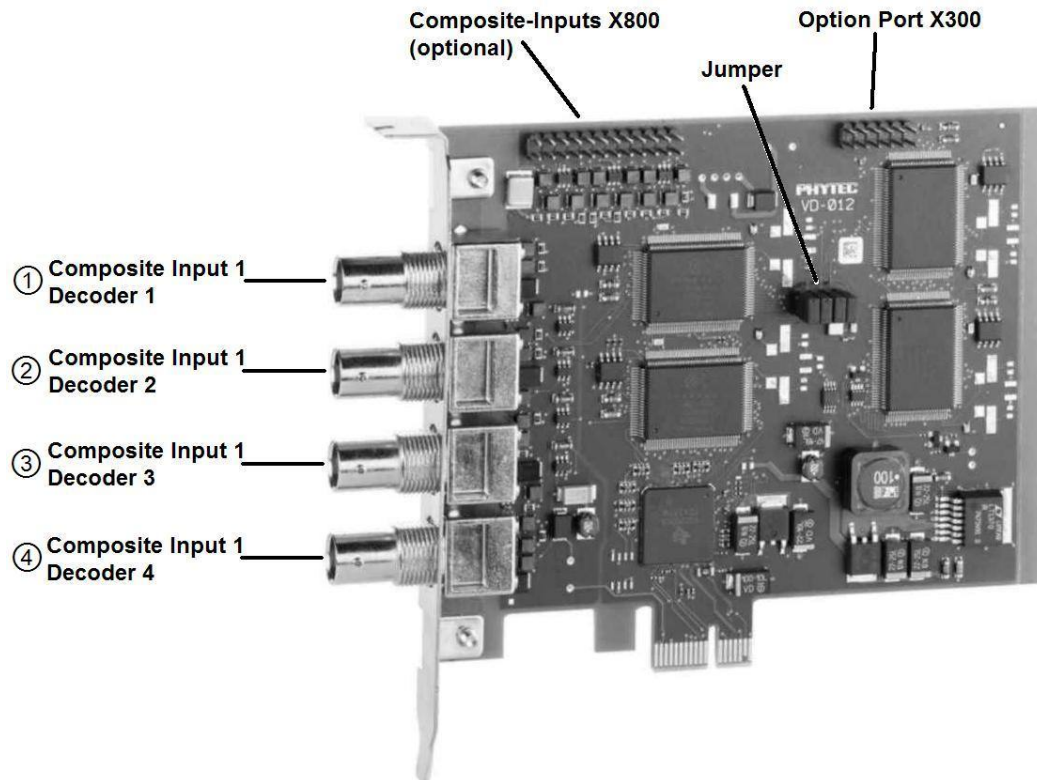


Figure 2: Connectors of the pciGrabber-4x4 (VD-012)

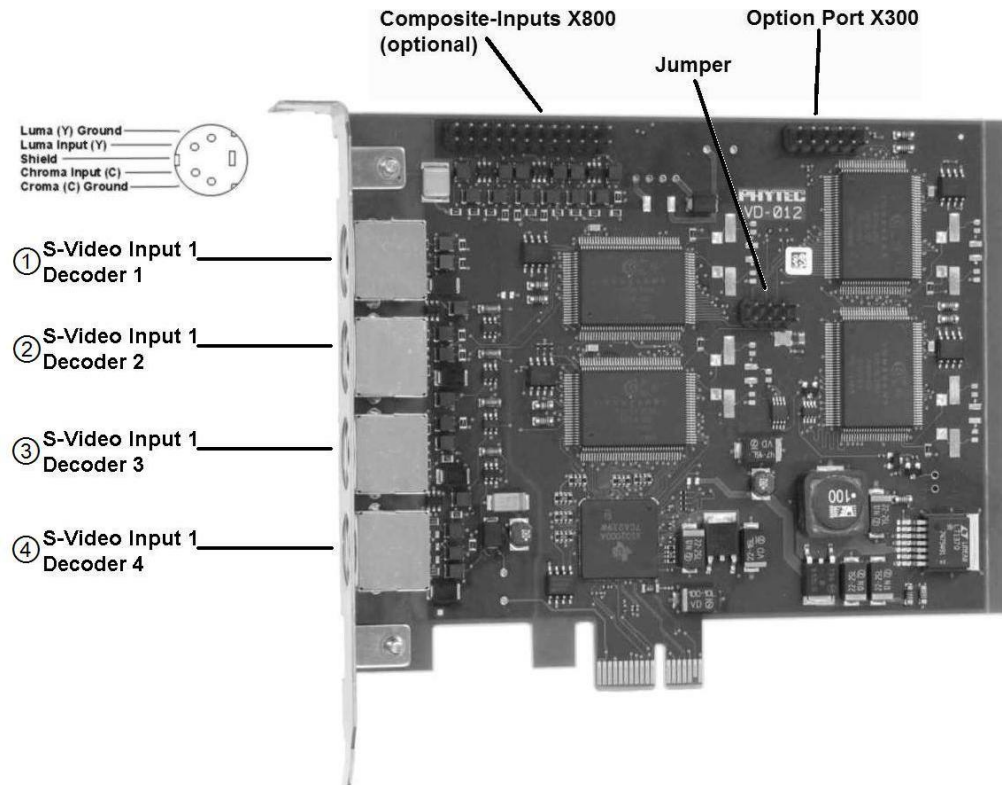


Figure 3: Connectors of the pciGrabber-4x4 (VD-012-X1)

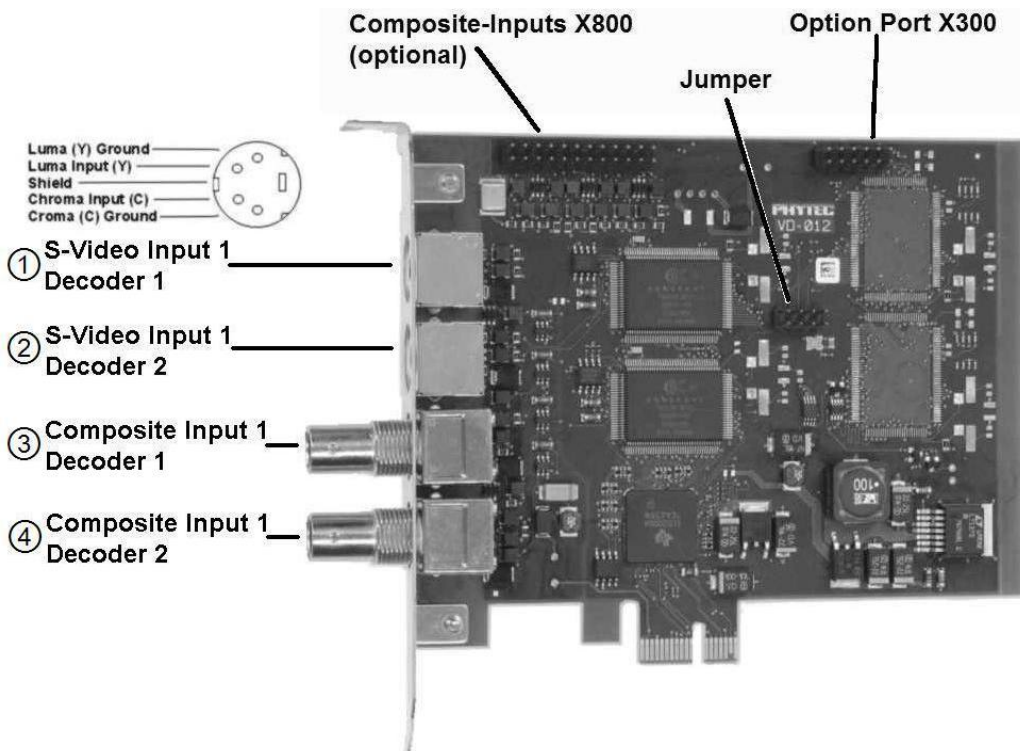


Figure 4: Connectors of the pciGrabber-4x4 (VD-012-X2)

### 2.5.1 Composite Inputs

All composite video sources with an output level of  $1 V_{ss}$  and an impedance of  $75 \Omega$  can be used. For more information on video standards, please refer to *section 2.2*.

- **Version VD-012**

Four composite inputs available to the Grabber are located on the four BNC sockets (Socket ①, ②, ③ and ④). On the Pin header row X800 are up to 12 additional inputs available. These inputs can be used with the extension Cards VZ-012 (see chapter 4.2).

- **Version VD-012-X1**

On the Pin header row X800 are up to 12 additional inputs available. These inputs can be used with the extension Card VZ-012 (see chapter 4.2).

- **Version VD-012-X2**

Two composite inputs available to the Grabber are located on the two BNC sockets (Socket ① and ②). On the Pin header row X800 are up to 4 additional inputs available. These inputs can be used with the extension Card VZ-012 (see chapter 4.2).

The input assignment for the channel numbers is as follows:

#### pciGrabber-4x4 (VD-012)

BNC ①	BNC ②	BNC ③	BNC ④
Composite Input 1 Decoder 1	Composite Input 1 Decoder 2	Composite Input 1 Decoder 3	Composite Input 1 Decoder 4

Table 1: Pin Assignments of the Model VD-012

#### pciGrabber-4x4 (VD-012-X1)

MINI-DIN ①	MINI-DIN ②	MINI-DIN ③	MINI-DIN ④
Composite Input 1 Decoder 1	Composite Input 1 Decoder 2	S-Vide Input 1 Decoder 3	S-Video Input 1 Decoder 4

Table 2: Pin Assignments of the Model VD-012-X1

**pciGrabber-4x4 (VD-012-X2)**

<b>BNC ①</b>	<b>BNC ②</b>	<b>MINI-DIN ③</b>	<b>MINI-DIN ④</b>
Composite Input 1 Decoder 1	Composite Input 1 Decoder 2	S-Video Input 1 Decoder 1	S-Video Input 1 Decoder 2

Table 3: Pin Assignments of the Model VD-012-X2

PHYTEC offers connecting cables to connect the application via BNC plugs and MINI-DIN plugs (see chapter 2.1).

### 2.5.2 S-Video Connection

The advantage of this design is the separate conduct of brightness and color signal. This prevents disturbing Moiré effects for fine image structures and improves the resolution of the color image.

S-Video sources can be connected to the variants VD-012-X1 and VD-012-X2. The pin Assignments can be taken from Table 2 and Table 3.

- The sockets are switched to the corresponding S-Video norms (refer to Figure 3 and Figure 4). The connection of the camera is possible using an S-Video cable.



## 2.6 I<sup>2</sup>C Interface

External devices can be polled or controlled via the I<sup>2</sup>C interface. In order for this to occur, the external devices must have an I<sup>2</sup>C interface operating in Slave mode.

The I<sup>2</sup>C interface is available at the internal pin header row X300 of the *Option Port*. It is possible to connect multiple I<sup>2</sup>C devices to the bus, but these devices must be distinguished by their device addresses. *Table 5* depicts the pin assignments for the pin header row X300.

X300	
Pin	Function
10	I <sup>2</sup> C Bus: SCL
12	I <sup>2</sup> C Bus: SDA
15	Ground

*Table 4 Connecting the I<sup>2</sup>C Interface to the Combi Socket*

### Note:

The maximum cable length is restricted, due to the fact that the I<sup>2</sup>C interface is driven by TTL signals. For a connected device, depending on the configured transmission rate, the maximum cable length is approx. 1 - 2 m.

Use cables with sufficient shielding when connecting this device.

Information for adapting the I<sup>2</sup>C interface into user software can be found in section 6.2.7, under the functions group „Transmitting Data via the I<sup>2</sup>C Interface“.

## 2.7 Jumper on pin header row X900

If more than one Framegrabber of the same variant are in one system it is not possible to discern them. Indeed they get different addresses but it is not possible to know which belong to whom. To solve that problem you have Jumper on the pin header row X900. You can use them to give every Framegrabber in one system a different address. So it is possible to discern them.

## **2.8 Notes on CE-Conformance and Immunity against Interference**

Upon delivery, the pciGrabber-4x4 meets all CE-requirements for household, office, manufacturing and industry. User modifications of the Grabber without permission of the manufacturer will result in the cancellation of the CE-certificate.

CE-conforming use of the Grabber is only maintained by utilizing CE-certified cables. These cables can be separately purchased from PHYTEC as accessories for the pciGrabber-4x4 (see chapter 2.1). If other cables are installed the user must ensure CE-conformity. If the S-Video cable WK-051 is used a ferrite of the type # 742 716 32 from Firm Würth, Kupferzell, Germany is required

If the user plans to connect the pciGrabber-4x4 with other cables, it is recommended that these cables are fitted with an anti-interference clamp or comparable interference suppression devices. The clamp should be placed about 5 cm from the Grabber and, the cable should be looped twice through the clamp.

For video cables a ferrite type # 742.711.4 from Firm Würth, Kupferzell, Germany is suitable.

The cable shielding has to be connected to the connector shell to obtain an optimum of shielding.

The pciGrabber-4x4 was tested for a standard PC environment. If the device should be used in a different environment, it has to be examined if additional radio shielding is necessary.

### **Caution:**

Please pay attention, that significant interference peaks (ESD) to the video signal or video ground might damage the input of the pciGrabber-4x4.

In areas with high interference level, for example in industrial areas and using long feed lines, additional precautions have to be taken to suppress interference. Long video cables, or mounting the components for image processing into plants and machines, can cause the exposition to balancing currents, which have to be eliminated from the

input of the pciGrabber-4x4 by appropriate arrangements. PHYTEC does not assume any liability for damages that occur due to incorrect connections of the signal source.

## 2.9 Option Port

The option port provides 8 digital I/O-lines and one I<sup>2</sup>C-interface to the user. The signals are routed to a connector with 6 x 2 pins. The connector is denoted as X300, pin 1 is located in the lower left. Figure 5 shows the assignment of the pins.

**Note:** The current drawn out of pin X300-1 (+5V) may not exceed 100 mA.

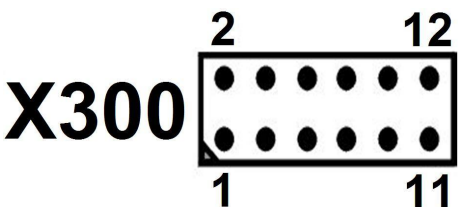


Figure 5: Pin Formation of the Option Port

### Option Port, X300

Erweiterungsanschluss (Option Port, X300)					
Pin	Funktion		Pin	Funktion	
1	+5V out		8	I/O 6	
2	I/O0		9	I/O 7	
3	I/O1		10	I <sup>2</sup> C SCL	
4	I/O2		11	I <sup>2</sup> C SDA	
5	I/O3		12	GND	
6	I/O4				
7	I/O5				

Table 5: Pin Assignment for the Option Port X300

### 3 Installation of the Grabber Card

The Grabber card converts analog signals from the camera and presents these signals in a digital form to the computer and software.

If you are not familiar with insertable cards, please take the time to familiarize yourself with the instructions and equipment. The following tasks are not difficult, but must be done with caution.

#### 3.1 Installing the Grabber Card

**Caution:**

The computer must be disconnected from the power supply. Please ensure that the device does not have any power supplied to it.

- Remove the housing of the PC (normally screwed).
- Select a free PCI Express slot  
(The free slots are normally the short white parallel slots on the motherboard).

*Please refer to the computer's mother board's User's Manual to obtain more information.*

- Remove the slot cover from the PC housing. The slot cover is located in front of the selected slot (unscrew or break off).
- As shown in Figure 6, insert the pciGrabber-4x4 into the slot with the connectors facing outwards. The card should be inserted securely.
- Do not force the card into the slot. Forcing the card into the slot can damage the mother board, as well as the card.
- Ensure that the Grabber card is inserted into the right PCI Express slot Line up the golden contact strips with the PCI Express slot's receptacle. Some resistance will be encountered as the contact strips spreads apart the contact springs.

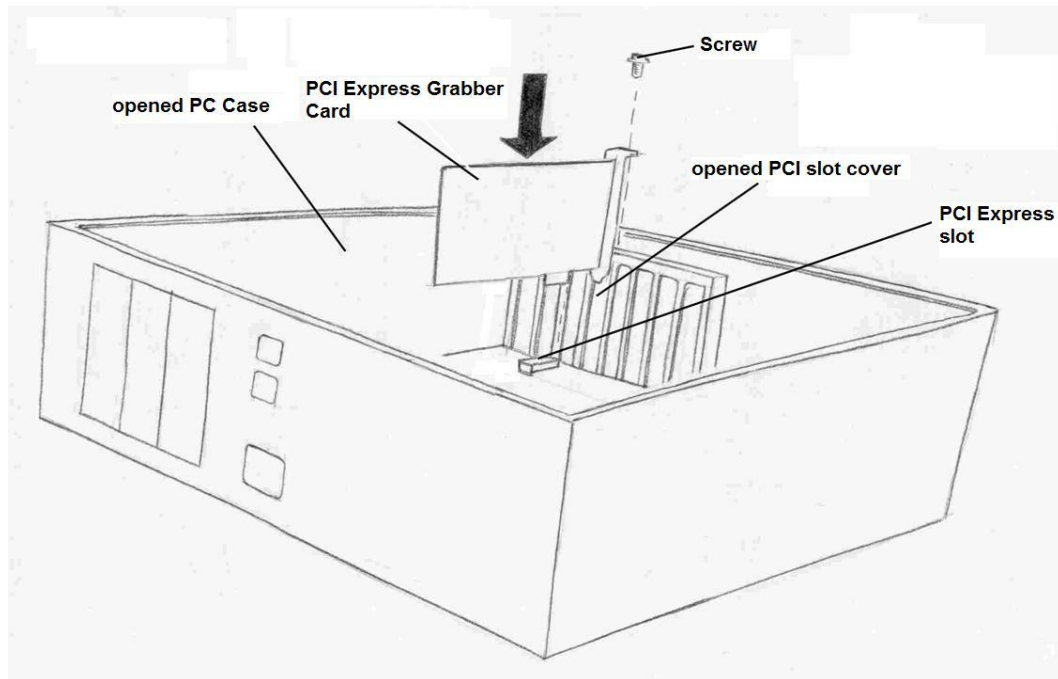


Figure 6: Inserting the Card into the PCI Express Slot

- After inserting the card, please ensure that the Grabber card fits snugly into the receptacle and that there is no interference from neighboring contacts.

**Caution:**

For stability reasons, and to ensure a secure Ground connection to the computer's housing, screw the card to the housing (see Figure 6)

- Close the computer's housing.

Next, the driver and the card's demo software must be installed.

Installing the driver's demo software differs depending on the operating system. The various installation procedures for installing the demo software is described in the next section 5.

### 3.2 Installing the Driver

- Connect the computer to the power supply and turn the computer on. During start-up the computer's BIOS should automatically recognize the card.

Two possibilities exist now:

1. Either the operating system recognizes the card and searches for the driver or
2. the operating system does not automatically recognize the card (i.e. Windows' NT) and the user must manually install the driver.

Depending on the type of operating system installed on the computer, installation occurs as follows:

- **Windows 2000/XP/VISTA™ :**

After the computer has recognized the card, the user is offered the option to install the driver.

Select the „*Search for the best driver for the device*“ option from the *Hardware Assistant* window,, and then confirm by selecting *OK*. In the next window that will appear, select *State a Position*. Now place the **PHYTEC Vision Utilities** CD into the CD-ROM drive. Select *Search*, and in the window that will appear, select the CD-ROM drive. Change the path to **pciGrab4driver\win2K\_98**. Confirm by selecting *OK*.

A list appears naming the drivers found on the CD. Select *PHYTEC PCI-Grabber* from the list.

The CD will automatically install the driver onto the computer.

Now the driver has been successfully installed.

Please *refer now to section 3*. Then *refer to section 5* to find information on how to install the demo software.

- **Windows<sup>®</sup> NT4.0<sup>™</sup> (with SercivePack 6):**

WindowsNT does not automatically recognize the card, therefore the driver must be installed manually. Place the **PHYTEC Vision Utilities** CD into the CD-ROM drive. From the main directory of the CD, select the program *Start.exe*, which is located under Windows<sup>®</sup> NT.

In the window that will appear, select the *PCI-Grabber*, and then select *Install drivers* and *WindowsNT4.0*.

After following the directions from the installation program, the necessary drivers will automatically be installed. In the window that will appear, confirm a Restart of the computer.

Now the computer should function normally after start-up of the operating system.

The driver has now been successfully installed.

Please *refer now to section 3*. Then *refer to section 5* to find information on how to install the demo software.

### **3.2.1 Additional Drivers (optional)**

It is possible to install additional drivers from the CD-ROM, although these drivers are not necessary for the functioning of the card described in this manual.

The **Twain driver** is a standard driver intended for use with graphic, photo, and scanner programs. The Twain driver reads images and works with the programs to process these images. The driver enables the Grabber and camera to function as a scanner device.

For additional information on the Twain driver, please *refer to the User's Manual on the graphic program that is being used.*

If installation of that driver is desired:

Place the **PHYTEC Vision Utilities** CD into the CD-ROM drive and start the file *start.exe*. This file can be found in the main directory of the CD.

In the window that will appear, select *pciGrabber*. An installation window will appear next containing the following two entries:

- ***Install Twain***



### 3.3 Installing the Demo Program

With a connected camera, the demo program allows the user to test the card, modify image parameters, and execute simple image operations.

To install the program:

- Place the **PHYTEC Vision Utilities** CD into the CD-ROM drive.
- The CD-ROM drive must be selected and the program *start.exe* (found in the CD's main directory) must be started.
- Select the *PCI bus grabber* from the install menu that will appear (see Figure 7).
- Click on *Install Windows demo software*.

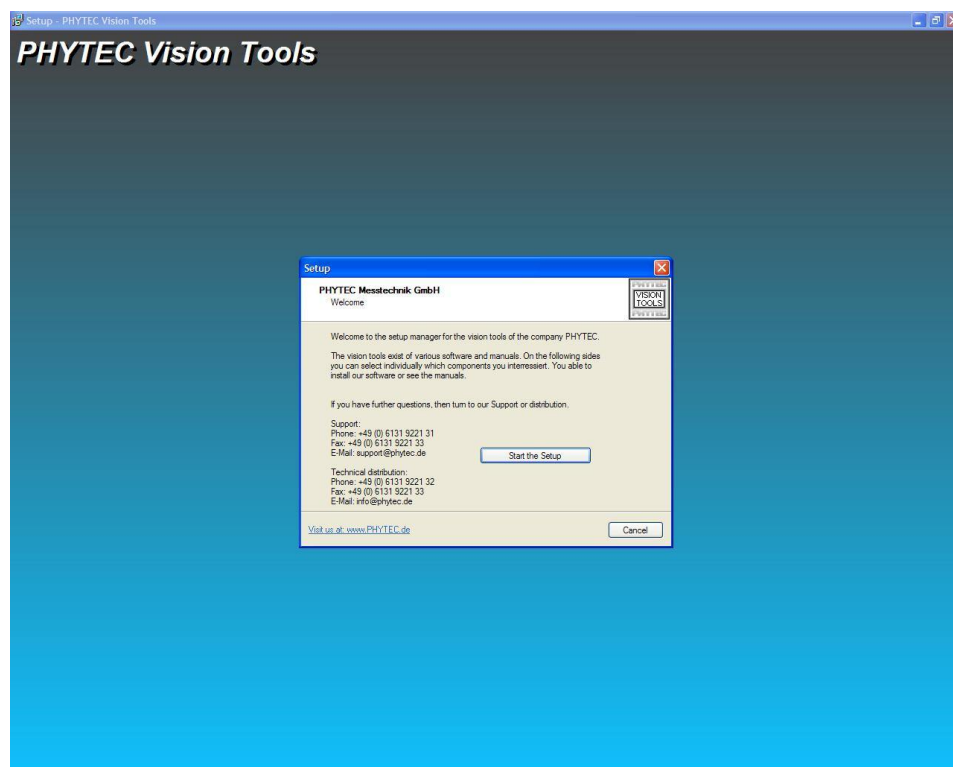


Figure 7: PHYTEC Install Menu

- Follow the installation instructions and the demo program will be automatically installed on the computer.

## 4 Connecting Video Sources

It is possible to connect one or more video sources to the pciGrabber-4x4 (see Figure 8, Figure 9 and Figure 10). These sources can either be video cameras, video recorders or any other video source [with appropriate outputs (composite or *S-Video*)].

Depending on the Grabber model, up to four composite (VD-012) or up to four S-Video sources (VD-012-X1) can be connected to the Grabber. At the VD-012-X2 can be connected up to two composite and up to two S-Video sources.

Changing channels occurs via software, or via the included demo program.

Per decoder it is possible to digitize one Channel. The variant VD-012 and VD-012-X1 works with four decoders and the variant VD-012-X2 works with two decoders.

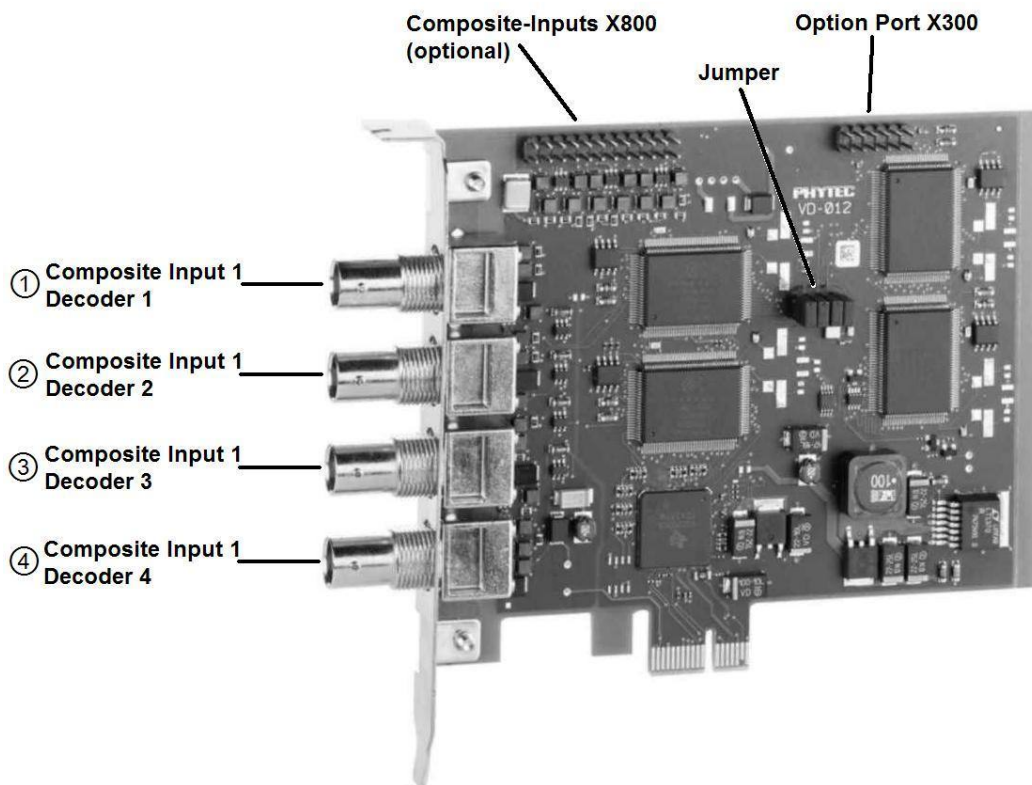


Figure 8: Overview of the pciGrabber-4x4 Connectors (VD-012)

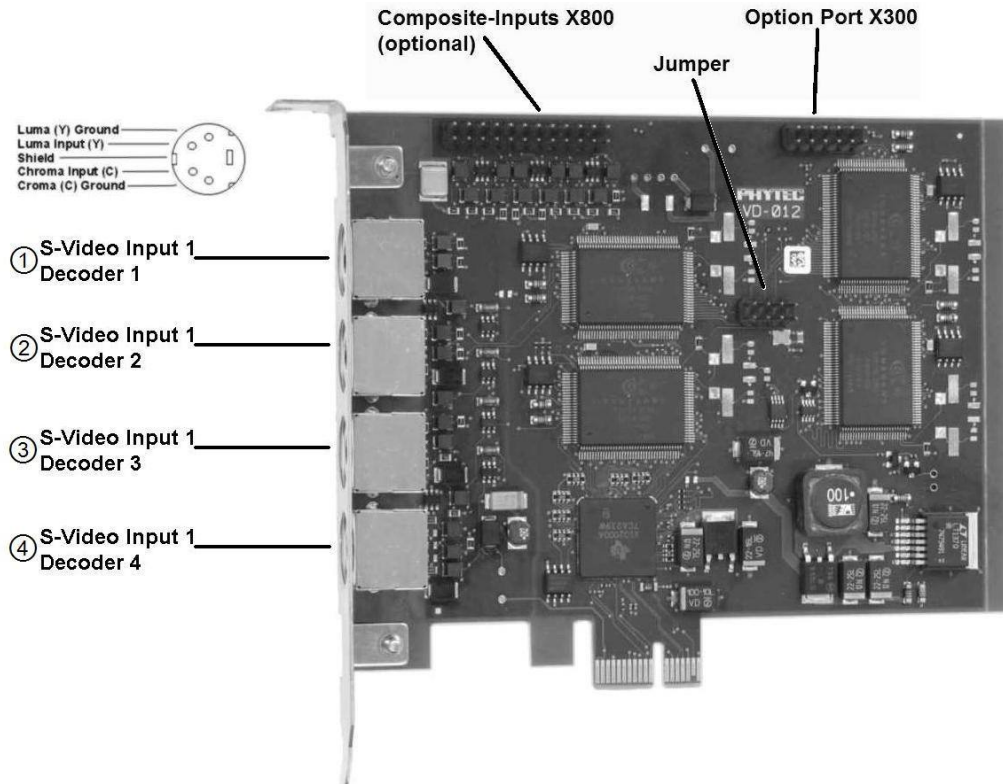


Figure 9: Overview of the pciGrabber-4x4 Connectors (VD-012-X1)

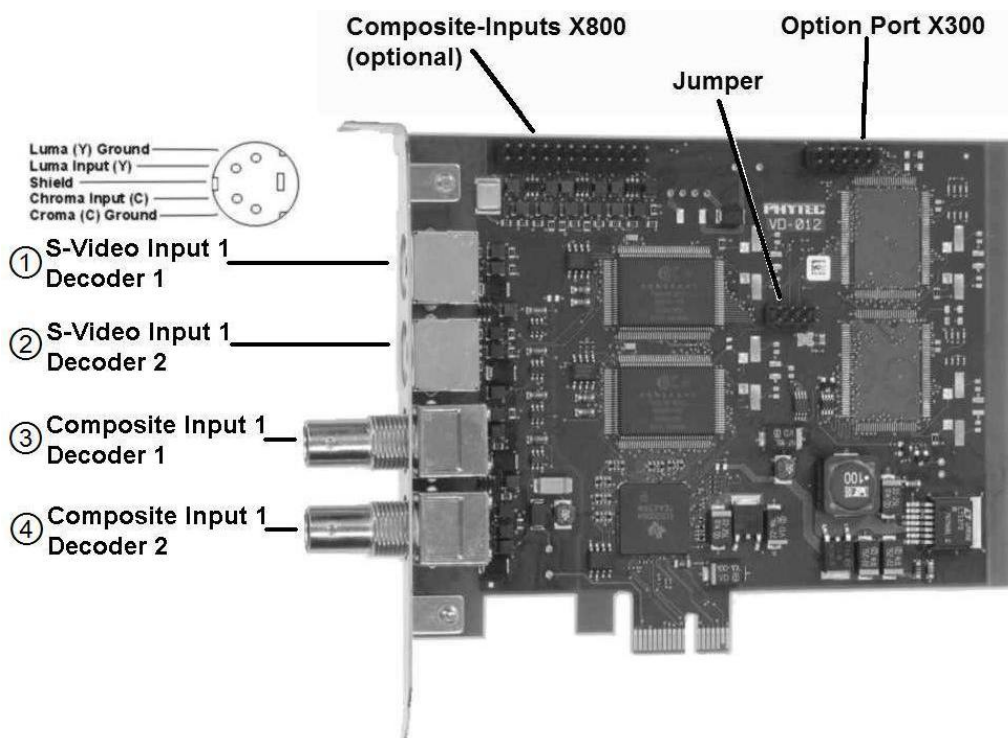


Figure 10: Overview of the pciGrabber-4x4 Connectors (VD-012-X2)

The composite inputs are located on the BNC sockets.

An S-Video signal can be applied to the Mini DIN sockets.

Necessary cables can be ordered from PHYTEC. Please *refer to section 2.1, "Accessories"*.

At the pin header row X800 are additional composite inputs (see chapter 4.2).

Precise information for the pin assignments of the sockets can be found in the *section entitled Technical Data*.

#### 4.1 Possible Video Connections

Various video source connections for the Grabber are briefly described in this section.

All of the pictured cables can be ordered from PHYTEC.  
The illustration of the cables includes a brief cable description and the PHYTEC order number (see the figure below).



*Figure 11: Video Connector Cables - (Description and PHYTEC Order Number)*

For more information on compatibility, please *refer to the video source User's Manual/Data Sheets*.

Connection possibilities vary according to the Grabber model.

The following images categorize the various Grabber models.

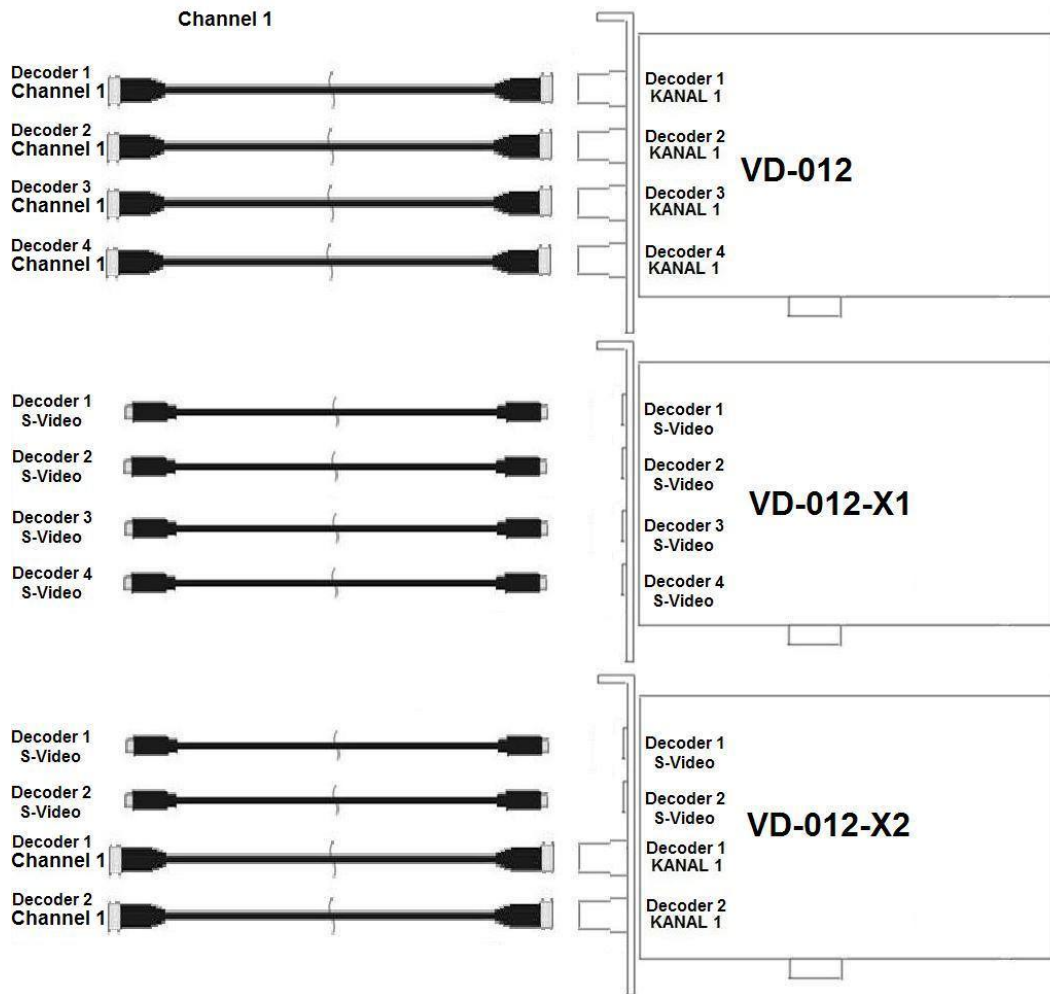


Figure 12: Connectors for the pciGrabber-4x4

The following section briefly describes the above depicted cables.

#### **4.1.1 The S-Video Cable**

The S-Video cable is connected to the Grabber using the round mini DIN socket. The video source to be connected (i.e. camera with S-Video output) should have a similar socket.

#### **4.1.2 The Composite Cable**

It is possible to connect the composite outputs (BNC plug) with a video source using a BNC plug.

**Note:**

If the composite sources contain a cinch socket, then a cinch/BNC adapter ( $75\ \Omega$ ) must be used.

In order to display an image, the correct channel must be selected in the user's software and in the demo program. It is possible for the included software to automatically recognize which channel is supplied with a signal (see section 5).

## 4.2 Extension Card VZ-012

An Extension Card for the pciGrabber-4x4 is available. With this Cards it is possible to use all the additional Video inputs on pin header row X800. In Figure 13 you can see the Card.

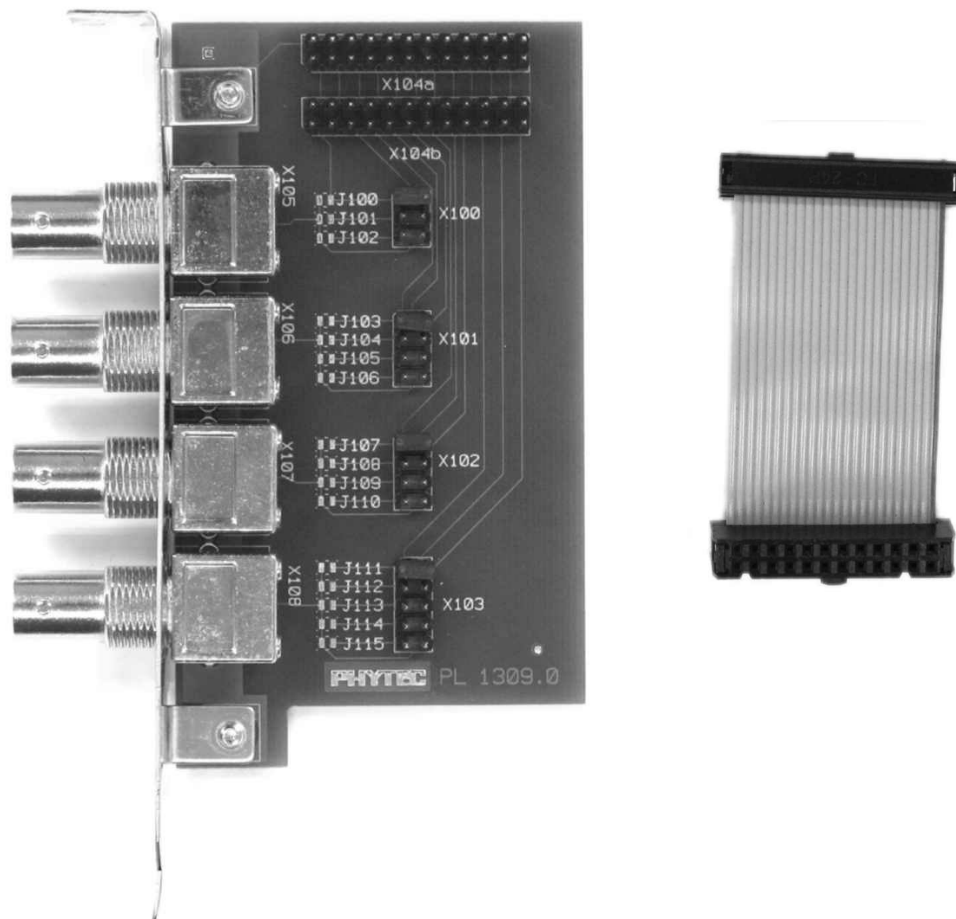


Figure 13: Extension Card VZ-012 with ribbon cable

**Dimensions:** 55 x 90 x 20 mm plus face plate and slot

The extension card have to connect with the framegrabber. The ribbon cable is connected to the pin header row X800 of the framegrabber. At the extension card can be select between X104a and X104b. It is

possible to operate several extension card parallel but it is important to set the right Jumper.

Every extension card offer four composite video inputs which are available on BNC sockets. The card can be placed into a standard PC.

In Table 6 is shown how many extension card can be used at the pciGrabber-4x4 variants.

	extension card 1	extension card 2	extension card 3
VD-012	•	•	•
VD-012-X1	•	•	•
VD-012-X2	•		

Table 6: Numbers of possible extension cards

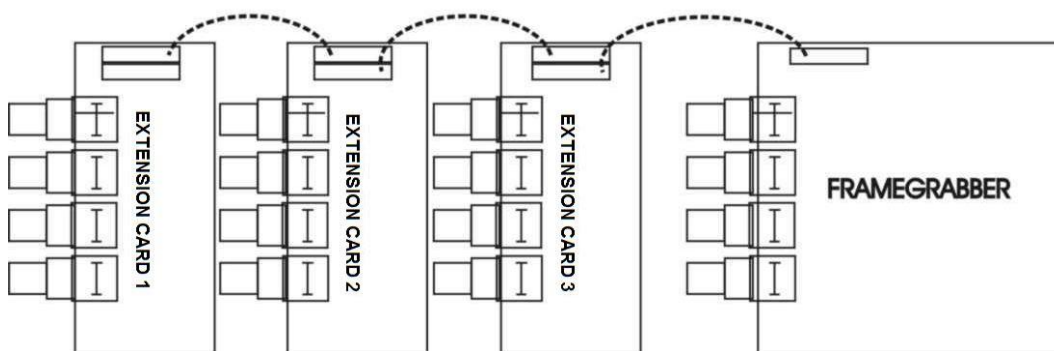


Figure 14: How to connect the extension cards

The extension card can be connected how in Figure 14. The first card is connected to the framegrabber and with the second extension card. The second is connected to the third card.

The right settings of the extension cards must be set with jumper. The settings are in Figure 15 and Figure 16 shown.



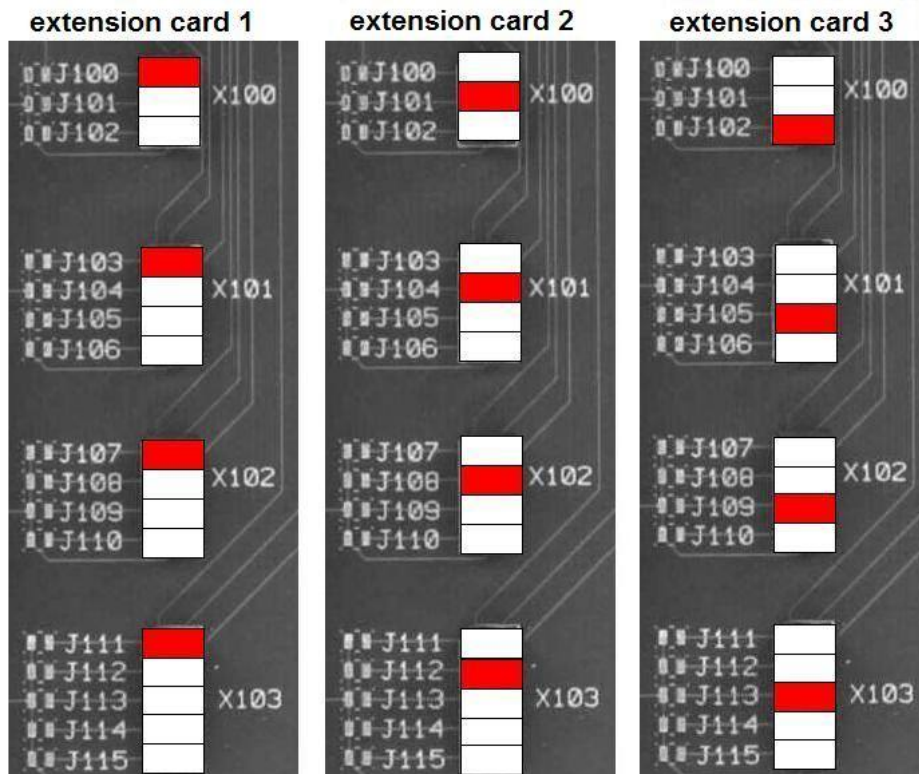


Figure 15: Jumper settings for three VZ-012 for VD-012 und VD-012-X1

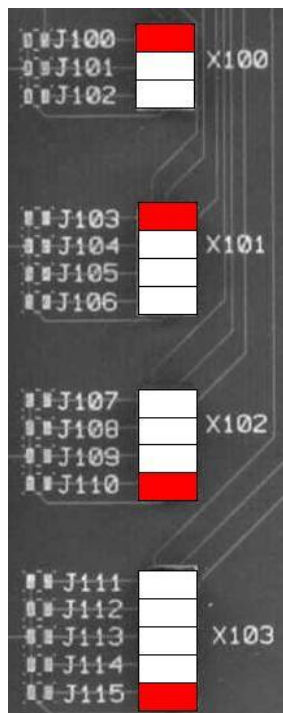


Figure 16: Jumper settings for VZ-012 for VD-012-X2

### 4.3 Overview about all video inputs

This chapter will give you an complete overview about all the video inputs inclusive the extension video inputs from the extension cards VZ-012. The jumper settings of the extension cards must be strictly adhered to the jumper settings in chapter 4.2.

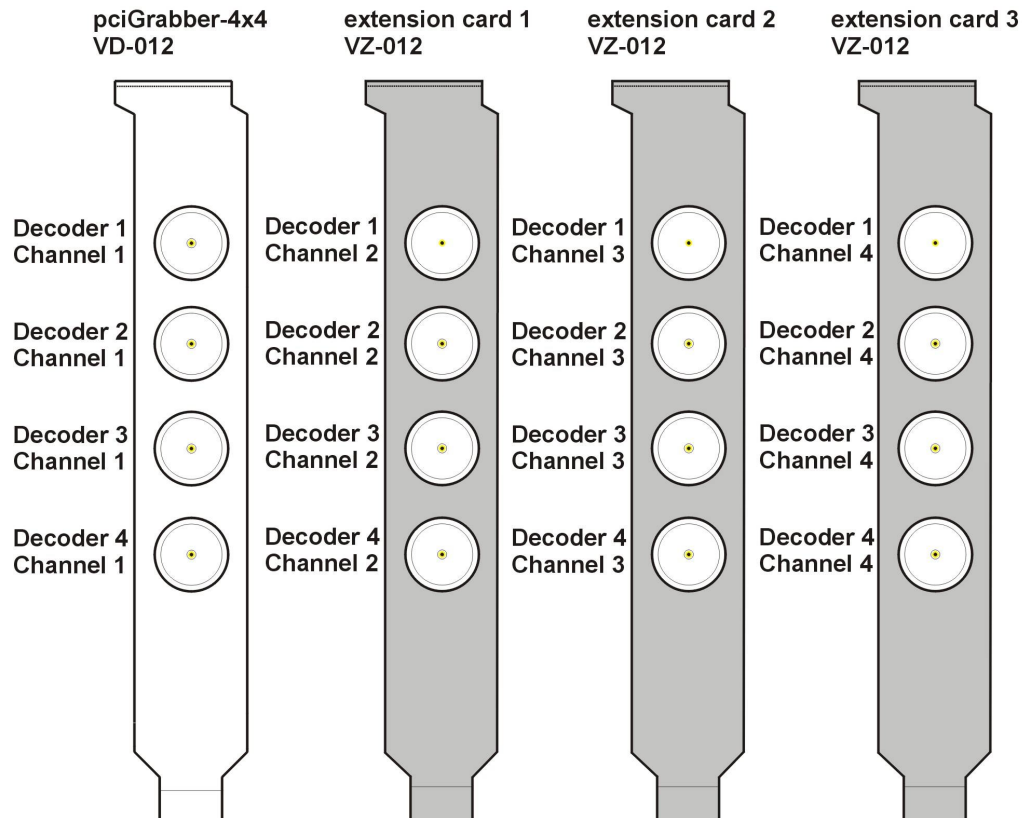


Figure 17: Video inputs VD-012

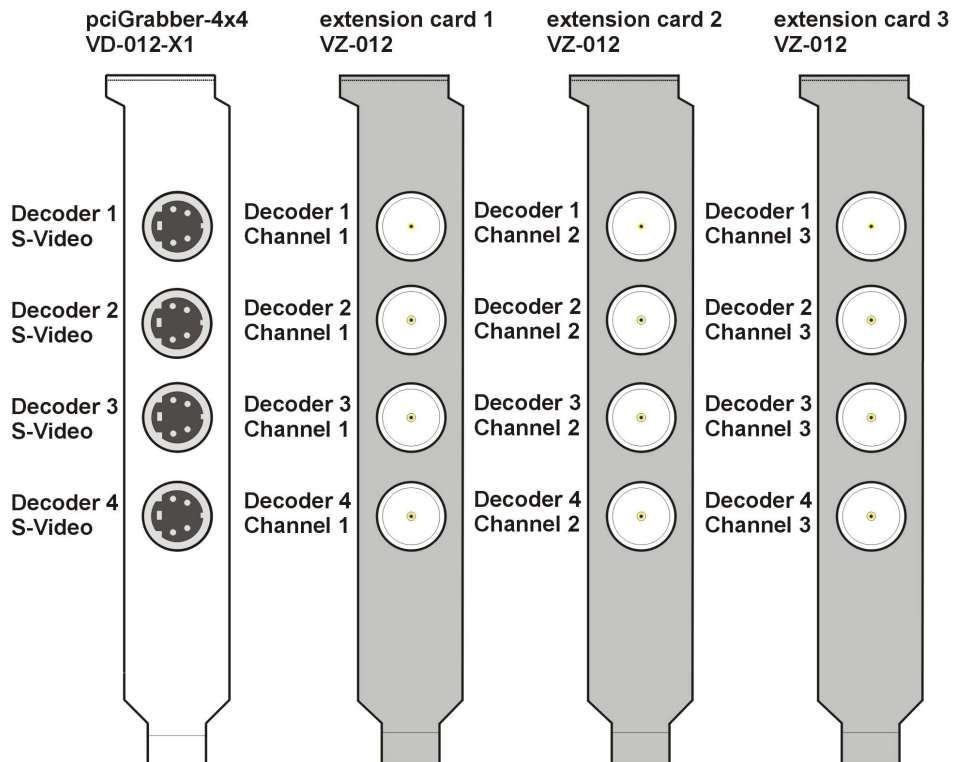


Figure 18: Video inputs VD-012-X1

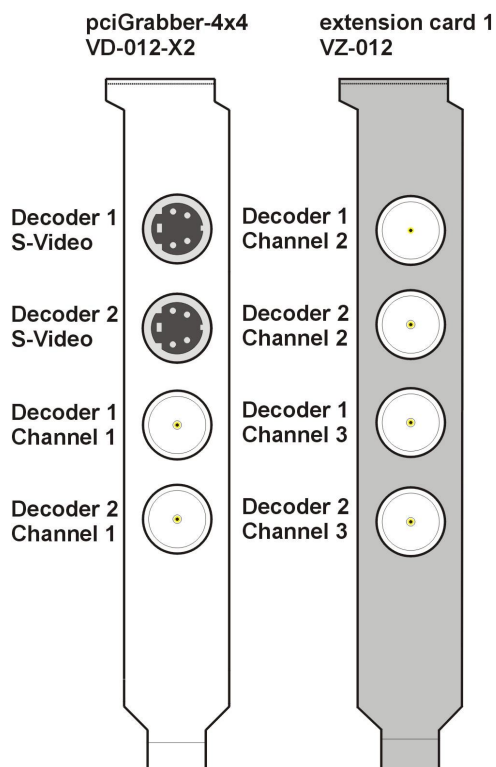


Figure 19: Video inputs VD-012-X2



## 5 Start-Up of the Grabber with Demo Programs

In order to continue with this section, the demo program and the Grabber driver must be correctly installed (see section 3).

The demo program can be found under *START / Programs / Phytex / pciGrabber4plus / Grab4PCI*. After this program has been started, an empty program window will appear with menu options (see Figure 20).

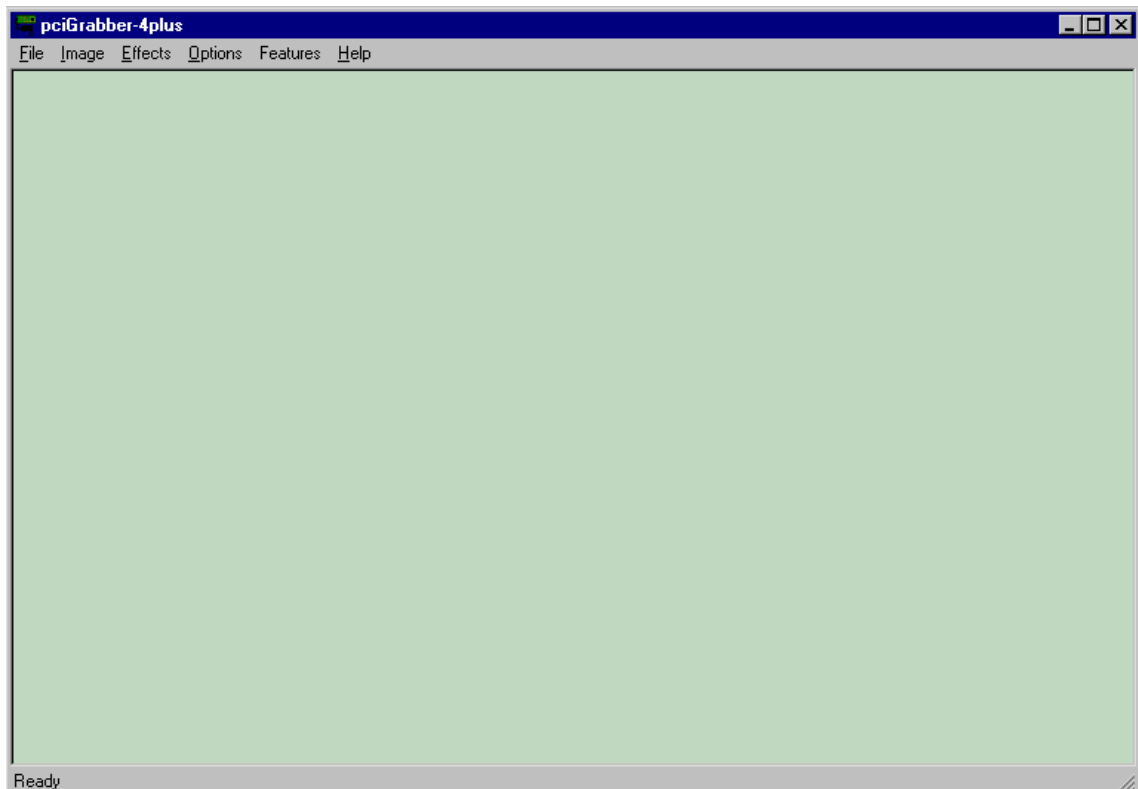


Figure 20: Overview of the Demo Program

Next, a moving live image from the camera should be displayed. Please ensure that a video camera, or another source is connected to the Grabber and that an image signal is being transmitted.

Basic parameters pertaining to the Grabber and arithmetic operations can be found in the *Options* pull-down menu.

*Basic Settings* contains the following menu:

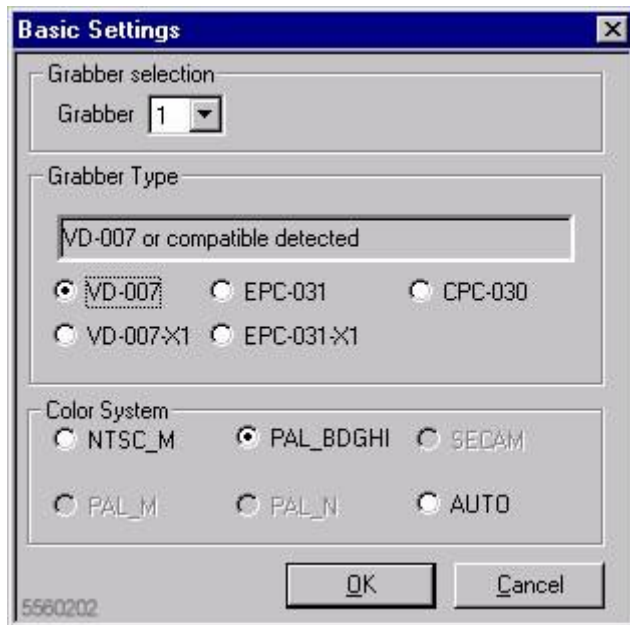


Figure 21: Basic Settings Menu

To select the decoder at the pciGrabber-4x4 the user must define which decoder the demo program is directed towards. Select the appropriate number in the *Grabber selection* field.

In the area *Grabber Type* is shown which Typ of Grabber and Decoder is installed.

The pciGrabber-4x4 appears as:

VD-012 Decoder 1  
VD-012 Decoder 2  
VD-012 Decoder 3  
VD-012 Decoder 4

or

VD-012-X1 Decoder 1  
VD-012-X1 Decoder 2  
VD-012-X1 Decoder 3

VD-012-X1 Decoder 4  
or  
VD-012-X2 Decoder 1  
VD-012-X2 Decoder 2

**Note:**

At the moment exists several PC-Framegrabber from PHYTEC.  
to this belong:

pciGrabber-4plus  
pciGrabber-4 express

These types are displayed as:

pciGrabber-4plus  
VD-009, VD-009-RS6, VD-009-X1 or VD-009-X1-RS6  
pciGrabber-4 express  
VD-011 or VD-011-RS6

When using an older PHYTEC Grabber model, the model is denoted as „VD-007 or compatible“. In this case, the exact type of card cannot be recognized and model VD-007 is automatically configured. To avoid this problem, select the installed Grabber from the list and configure it manually (i.e. VD-007 or VD-007-X1).

*Color System* configures the color system to be used with the Grabber.

PAL is mainly used in Europe and NTSC is used in the USA.

*Grabber Type* displays the recognized Grabber model.

While operating with live images, these parameters cannot be changed.

*Addition Settings* and *Type Casting Settings* are described with *Add Live Images* and *Arithmetics* later on in this manual. All of these entries can be found under the menu option *Features*.

Click on the *Image* button and the following pull-down menu will appear (*see below*).

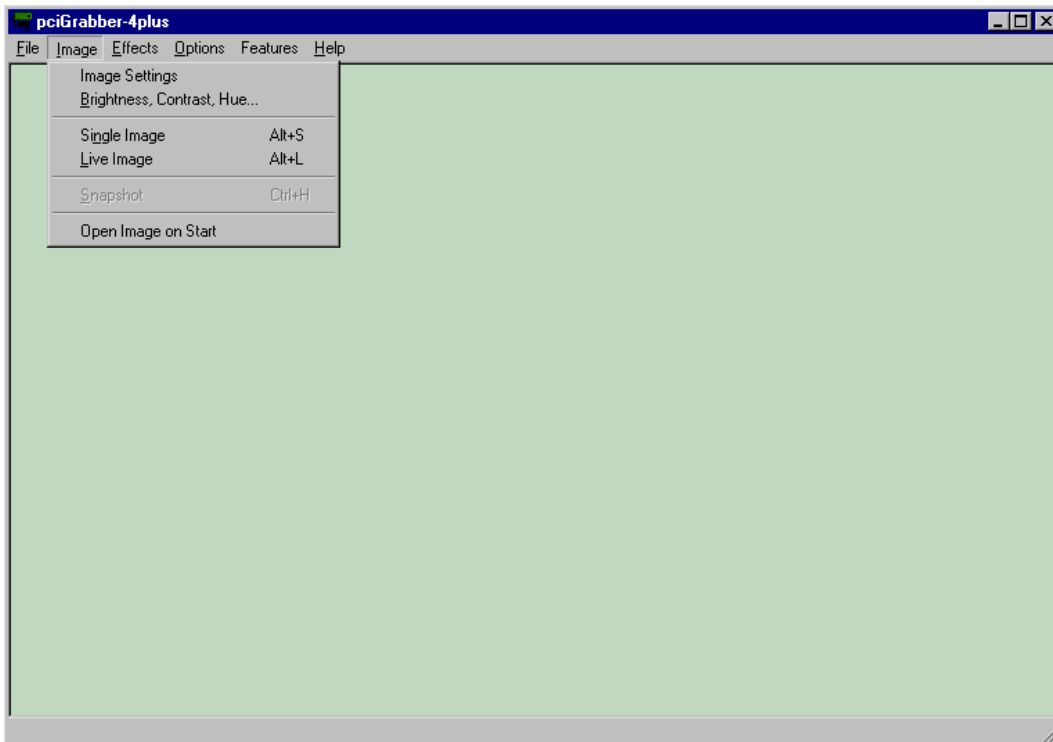


Figure 22: Menu Option: Image

In order to configure the parameters of the image to be grabbed, select the *Image Settings* command from the pull-down menu (see Figure 23).



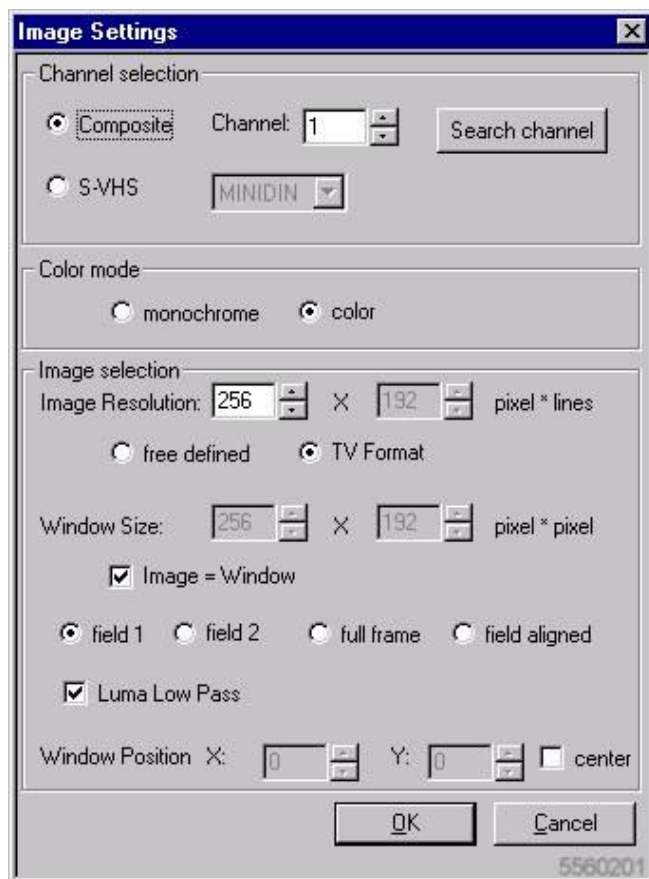


Figure 23: Configuring the Image Parameters

Detailed descriptions of each parameter will be given further on in this manual. In order to test the Grabber, the live image should be displayed on the monitor. To display the image on the monitor, the following requirements must be met.

It is important to select the proper video input for the Grabber. In the *Channel selection* field, fill in the type of video source (composite/ S-VHS) and the input channel that is being used.

The input channels can either be manually entered, or automatically searched for. In order to use the automatic search, click on the *Search channel* button. The first channel with an active video source that is found is used.

Depending on whether decoder is chosen under *Basic Settings* you must connect the camera to corresponding socket to get an picture.

	VD-012 Decoder 1	-> Channel 1 at socket ①
	VD-012 Decoder 2	-> Channel 1 at socket ②
	VD-012 Decoder 3	-> Channel 1 at socket ③
	VD-012 Decoder 4	-> Channel 1 at socket ④
or		
	VD-012-X1 Decoder 1	-> MINIDIN at socket ①
	VD-012-X1 Decoder 2	-> MINIDIN at socket ②
	VD-012-X1 Decoder 3	-> MINIDIN at socket ③
	VD-012-X1 Decoder 4	-> MINIDIN at socket ④
or		
	VD-012-X2 Decoder 1	-> Channel 1 at socket ①
		-> MINIDIN at socket ③
	VD-012-X2 Decoder 2	-> Channel 1 at socket ②
		-> MINIDIN at socket ④

In the *Color Mode* field, the user can choose to display the image in color, or in monochrome.

The remaining entries under *Image selection* can retain their pre-configured values.

Exit the menu by clicking *OK*.

Now select the *Live Image* command from the *Image* pull-down menu.

A live image from the selected video source will now be displayed in a new window (see Figure 24)

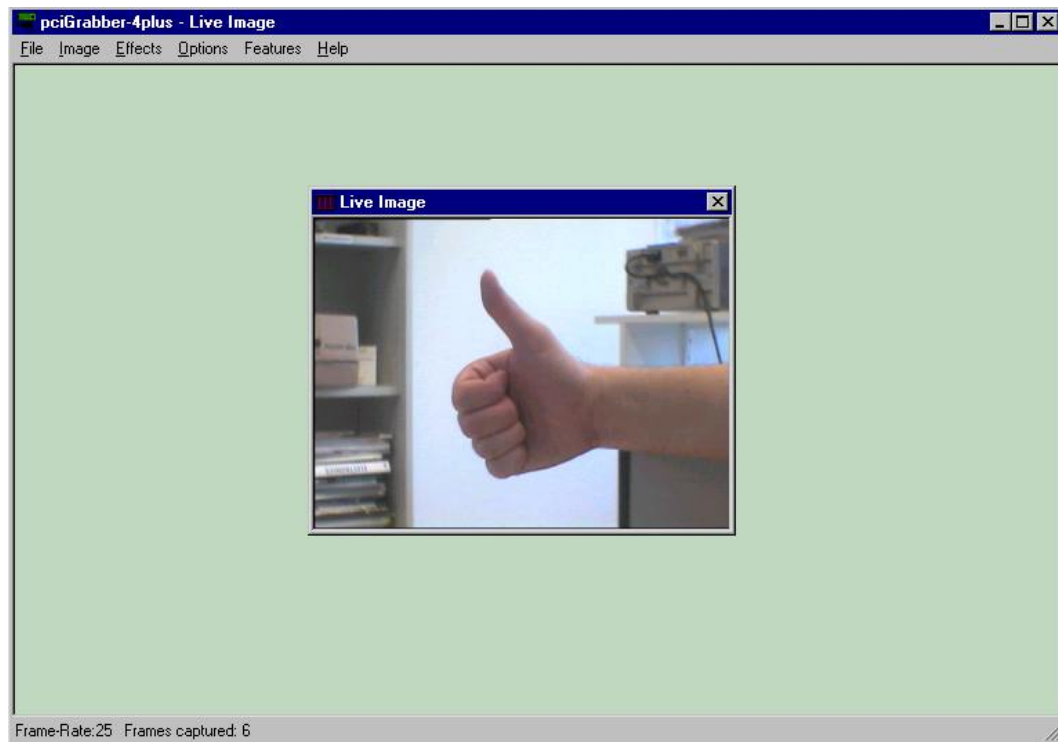


Figure 24: Live Image from the Video Source

If a blue screen appears, examine all connections to ensure that they are secure. Also ensure that the camera is receiving power.

If the connections are secure and a power supply is available, then perhaps an incorrect channel or Grabber was selected.

Additional error source are described in the appendix.

**Note:**

When operating multiple Grabbers in a computer, the user must select a primary Grabber. Designating a Grabber can be done under *Options*.

The *Frame Rate* display *xx* (*xx*= Number) can be found on the lower bar of the main window. The value represents the number of images that are generated per second in the live window. The value is dependant on the size of the image, and the capacity of the computer, because the digitized image must be transmitted from the computer's RAM to the graphic card to eventually show up on the screen.

**Note:**

Despite the processor's capacity, the Grabber always stores image data in real time in the main memory (RAM) of the PC.

Further processing of the data is dependant on the CPU of the PC.

The status bar further contains a counter that displays the total number of live images captured (*Frames Captured*).

When the counter has reached 255, it automatically begins a new sequence starting with 0.

The status bar can also be used to indicate whether the Grabber is active or not.

## 5.1 The parallel image processing

The demo program does not supported the parallel image processing, but it is possibility to open the program several times. If you use a pciGrabber-4x4 variant with four decoder you can start the demo program four times.

First the settings in the menu *Basic Settings* must be set. It is important to allocate every demo program one decoder. If two programs have the same decoder it is possible that they work incorrect.

After that you must select the channel under *Image Settings*. Finally the *Live Image* can be started.

The following should be noted! The pciGrabber-4x4 uses a PCI Express-to-PCI-Bridge. This means that all video decoder operates together on a PCI bus. The PCI bus works only on the grabber card. Through the bridge the PCI bus is linked to the PCI Express bus. The PCI Express bus works with 1250 MHz and is able to transmit about 250 MB/s. The PCI bus works with 33 MHz. So it is possible to transmit 132 MB/s. So the PCI bus limits the data volume. At a digitizing of one decoder the data volume represents 44 MB/s.

- The calculation of the maximum data volume per one decoder results (PAL):

Image size (max): 768 x 576

Numbers of bytes per pixel(RGB32): 4

Numbers of pictures pro second: 25

Numbers of decoders: 1

$$size \cdot \frac{byte}{pixel} \cdot \frac{pictures}{s} \cdot decoders = \frac{MByte}{s} \quad \text{equation 1}$$

$$768 \cdot 576 \cdot 4Byte \cdot 25 \frac{1}{s} = 44.2368 \frac{MByte}{s}$$

Are four decoders used simultaneously, increasing the volume of data by a factor of four to 176.9 MB/s

$$768 \cdot 576 \cdot 4Byte \cdot 25 \frac{1}{s} \cdot 4 = 176.9 \frac{MByte}{s}$$

To get a clear transfer of the data volume the maximum data rate of the PCI bus must not be exceeded. In addition, a safety distance of ~15% to the 132 MB/s must be adhered to. So the maximum data without overloading the PCI bus is 112 MB/s

Numbers of decoders	RGB32 768 x 576	RGB32 640 x 480	RGB32 620 x 450	RGB16 768 x 576	Y8 Gray 768 x 576
1	<b>44,2 MB/s</b>	<b>30,7 MB/s</b>	<b>27,9 MB/s</b>	<b>22,1 MB/s</b>	<b>11,1 MB/s</b>
2	<b>88,5 MB/s</b>	<b>61,4 MB/s</b>	<b>55,8 MB/s</b>	<b>44,2 MB/s</b>	<b>22,1 MB/s</b>
3	<b>132,7 MB/s</b>	<b>92,2 MB/s</b>	<b>83,7 MB/s</b>	<b>66,4 MB/s</b>	<b>33,2 MB/s</b>
4	<b>176,9 MB/s</b>	<b>122,9 MB/s</b>	<b>111,6 MB/s</b>	<b>88,5 MB/s</b>	<b>44,2 MB/s</b>

Table 7 Example data volumes PAL 25fps

Numbers of decoders	RGB32 720 x 480	RGB32 640 x 480	RGB32 580 x 400	RGB16 720 x 480	Y8 Gray 720 x 480
1	<b>41,5 MB/s</b>	<b>36,9 MB/s</b>	<b>27,8 MB/s</b>	<b>20,7 MB/s</b>	<b>10,4 MB/s</b>
2	<b>82,9 MB/s</b>	<b>73,7 MB/s</b>	<b>55,7 MB/s</b>	<b>41,5 MB/s</b>	<b>20,7 MB/s</b>
3	<b>124,4 MB/s</b>	<b>110,6 MB/s</b>	<b>83,5 MB/s</b>	<b>62,2 MB/s</b>	<b>31,1 MB/s</b>
4	<b>165,9 MB/s</b>	<b>147,5 MB/s</b>	<b>111,4 MB/s</b>	<b>82,9 MB/s</b>	<b>41,5 MB/s</b>

Table 8 Example data volumes NTSC 30fps

If the 112 MB/s are not exceeded an unproblematic data transfer is possible. To see as in Table 7 and Table 8 it exists many possibilities to reduce the volume of the data.

1. *Reduce the resolutio*
2. *Reduce the colour depth*
3. *Reduce the display regeneration rate*

With the help of equation 1 can compute the data volume and decide which settings for the suitable application suits.



*Figure 25: Overloaded PCI bus*

Figure 25 shows an effect which can appear with too high data rate. It is good to see that the whole picture becomes streaky. The stripes arise from the fact that some picture data in absence of the capacity of the PCI bus can not be written fast enough in the main memory. The data which cannot be transmitted get lost, because the video decoder cannot cache them. Only the “old data” are available for certain ranges of the pictures. In these ranges the “old picture data” are displayed and so the picture becomes streaky.

## 5.2 Demo Program Description

This section describes in greater detail the program, as well as the menus of the included demo program.

The *Image Settings* menu (see Figure 26) contains parameters that influence image generation and depiction:

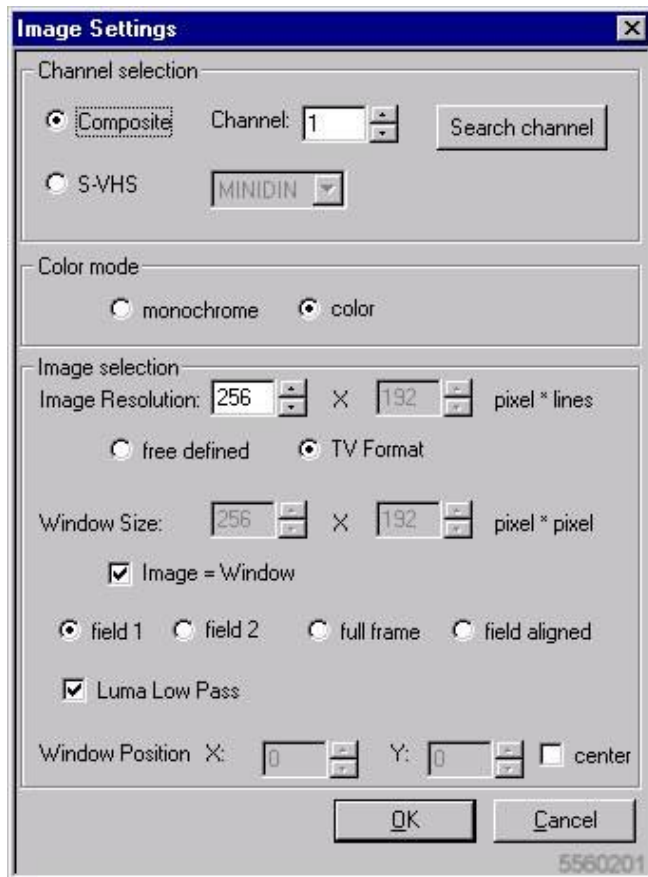


Figure 26: „Image Setting“ Menu

The parameters can be configured before a live image is displayed, although parameters cannot be configured while live images are displayed.

The section entitled *Channel Selection*, offers parameters for video source types and channel selection.



Click on either the *Composite* or the *S-VHS* button to select the appropriate signal type.

- **Composite Sources**

*Composite* refers to the BNC sockets (VD-011).

From the *Channel* menu, select the appropriate input channel for the connected camera. Note that the right decoder is selected.

Clicking on *Search Channel* allows the grabber to search for an active input channel. The program configures the first channel with a video signal. This function can only search on one decoder. To check the whole card every decoder must select separately select under *Basic Settings*.

By using the extension boards VZ-012 16 video inputs are available.

- **S-Video Source**

*S-Video* (or „S-VHS“) – Sources become designated with MINIDIN. The image source is connected to the round mini DIN socket.

The user can choose to display an image in color (when using a color video source) or monochrome by using the *color* and *monochrome* buttons.

“*Image Selection*“, found in the lower section of the window, can be used to configure the size and resolution of the image.

The *Image Resolution* parameter is used to configure the image's resolution (= „quality“)

The parameters divide into x-direction for the pixel number and in y-direction for the row number. Both values can be changed separately using the *free defined* button.

Please note, that the image will be displayed distorted (stretched or shrunk) if the 4:3 ratio is not adhered to. (This width to height ratio arises due to television standards).

The *TV Format* button prevents image distortion by automatically adhering to the 4:3 ratio (width/height relationship). For example, if given the number of pixels, the number of rows is automatically calculated with the 4:3 ratio.

The *Window Size* button can be used to extract a section of the image, and display this section instead of the whole picture on the monitor.

This section can be smaller than the viewing field of a camera. If the entire digitized field is to be displayed, then checkmark the *Image=Window* box.

The *Window Size* does not distort the image geometry because it is not a scaled section, rather than a cut out section.

**Note:**

Please note that scaling and cutting section processing is run in real time in the Grabber. The Grabber stores the image as it is displayed on the monitor. This is very beneficial because the CPU is not needed for this function.

A brief explanation of similar television technology will lead to a better understanding of the buttons *field1*, *field2*, *full frame*, and *field aligned*.

A television image (normal video signal) is made up of two interlaced images, so called half frames (fields) (see Figure 27). These half frames (fields) are consecutively generated in a similar fashion and then displayed on a screen (i.e. television).

The interlacing of the images reduces the flickering that can occur with TV images.

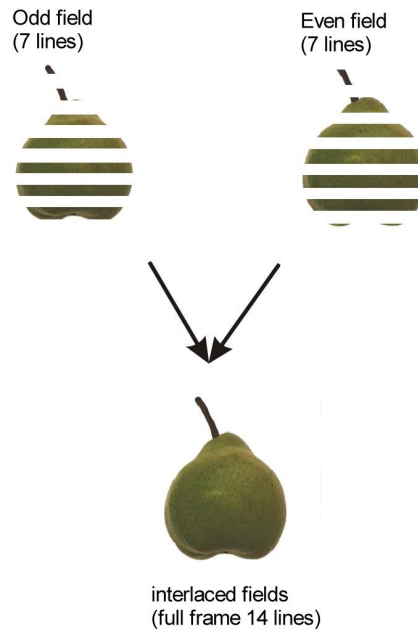


Figure 27: Creating a Full Image: Two Fields, Each with 7 rows

According to the PAL-norm, each signal contains 625 rows. The rows are divided into field frames: the first field (*odd, field1 with rows 1-625*) and a second field (*even, field2 with rows 2-624*). An image section is fully recognizable by one of its half fields. The image's vertical resolution is reduced by half, since the image is only represented by 228 rows. (excluding the invisible rows that precede and succeed the image as well as test and data rows.) A total of approximately 576 from 625 rows remain visible

Digitizing a field is time efficient; compare 20 ms for a field image to 40 ms for both fields (*full frame*).

If the same field (i.e. the first) needs to be digitized repeatedly, there is a pause of 20 ms between the processes.

Digitizing a full frame can create a distorted image if the object moves too quickly. The object is in a different location in the first field than it is in the second, creating a comb effect. The image may appear as shown in Figure 28.



Figure 28: Comb Effect That Occurs with Quick Moving Objects

The parameter described above can be changed in the demo program.

With vertical resolutions smaller than 288 rows, it is easier to digitize a field. The *field1* (first, odd half frame) and *field2* (second, even half frame) buttons can be used to manipulate the digitization of half frames.

If the number of rows is larger than 288, then both fields must be digitized. To digitize both fields, click on *full frame*. If a number larger than 288 is entered into *Image Resolution*, then the *full frame* is automatically selected.

The *field aligned* button doubles the number of displayed half frames per second. This eliminates the 20 ms pause between the digitization of half frames.

Optically frames, the image contents shifts a half line up and down, when consecutively displaying both half frames.

This occurs because the two half frames cannot be interlaced to form a full frame. When configuring *field aligned*, the Grabber automatically moves the second half frame one half row, so that the second half frame can properly interlace with the first half frame, creating a full frame.

This does not allow the jump effect to occur. This configuration for *field aligned* is also helpful when the user wishes to consecutively digitize images with a maximum of 288 rows, at a rapid pace (1 field in approx. 20 ms).

If the horizontal resolution is smaller than 360 Pixel the checkmark came Lowpass should be set.

This Lowpass will smoothen the in size reduced image. The Lowpass will automatically be activated if the resolution is smaller matically be activated if the resolution is smaller then 360 Pixel and otherwise deactivated.

*Window Position* can be used to determine the position of the image window contained in the above mentioned image section. The values represent the position of the upper left-hand corner. In order to center the image in the TV screen, check mark the box next to *center*.

The parameter moves around cut out the section in the an entire image. Therefore, can the cut out section only be moved around if it is smaller than the entire image.

### 5.3 Image Control

During the displaying of a live image the image control window can be opened by selecting *Image-Live\_Image Control*. The dialog box shown in Figure 29 will appear. With the help of slider controls the values of brightness, contrast, color, saturation and hue are adjusted. The values are immediately applied to the Grabber, so that the corresponding effects can be registered in the live image.

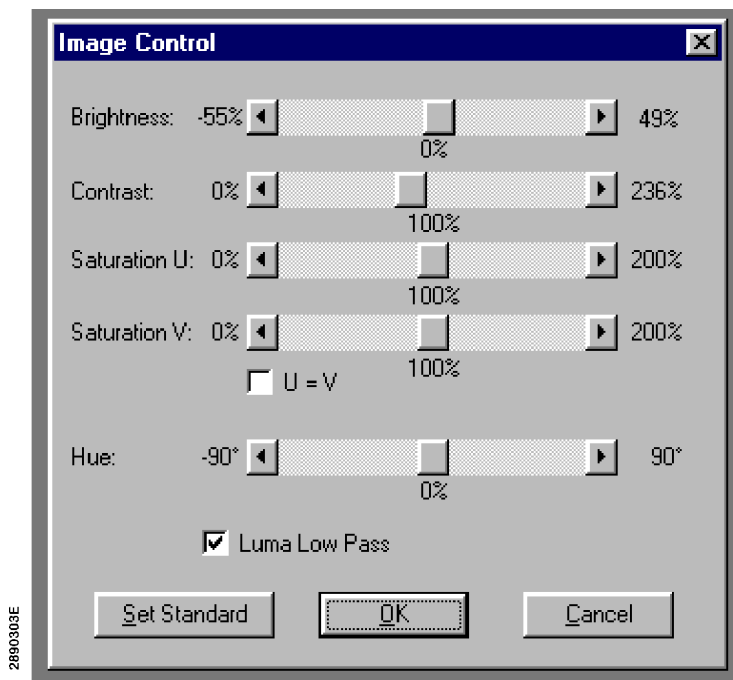


Figure 29: The Image Control Window

For the adjustment of the color saturation two controls are available: saturation U and saturation V. This allows separate manipulation of the saturation in the red- and blueviolet region. With the control box  $U=V$  both controls can be united. In this way you are able to change the color saturation without manipulating the color tone.

The *hue* control makes only sense for the NTSC-system. This control serves for the correction of the color tone, in case a phase shift has occurred during transmission. Those interference's can only be present in NTSC-systems. The PAL-system corrects color tone failures automatically, so that the hue control has no effect.

**Note:**

Modification of the hue (i.e. white balance) can equally be done on NTSC and PAL systems by moving the saturation sliders separately (in general it is better to midify the white balance at the camera or video source if possible).

## 5.4 Additional Functions Under *Image*

- Using the ***Single Image*** entry, a snapshot is taken and displayed on the screen. In this mode, the Grabber only performs one digitization.
- The parameter *Image Settings* defines the image.
- Using the parameter ***Live Image***, a live image can be displayed on the monitor. *Image settings* also defines the image in this mode.
- Snapshots can be taken during live operation using the ***Snapshot*** option.
- The current image will be displayed in a new window. Multiple snapshots can be made.
- Snapshot-Windows that appear on the screen are automatically numbered.
- Using the pre-configured parameters in *Image Settings*, ***Open Image on Start*** enables a live image from the video source to be displayed on the monitor after every program start.
- Adding the demo program to the auto start group enables the computer, after start-up, without intervention from the user, to display a live image, with the pre-configured parameters, on the monitor.

## 5.5 Crosshair function (Overlay)

Several types of crosshairs can be overlayed in the live image. This can be useful to center an object in the middle of the image.

The parameters for this function can be found under the menu option *Effects*. All of the cross hairs, or a combination of them, can be overlayed in the image.

## 5.6 Special Functions

The demo program offers several special functions to manipulate and analyze image contents.

- **Display Histograms**

*Histogram* enables a histogram to be calculated from a static image, i.e. an image obtained using the *Snapshot*.

A histogram provides the distribution of the grey- or color values of an image.

The relative Frequency of the corresponding intensity values are represented by brightness, as well as the intensity (see Figure 30).

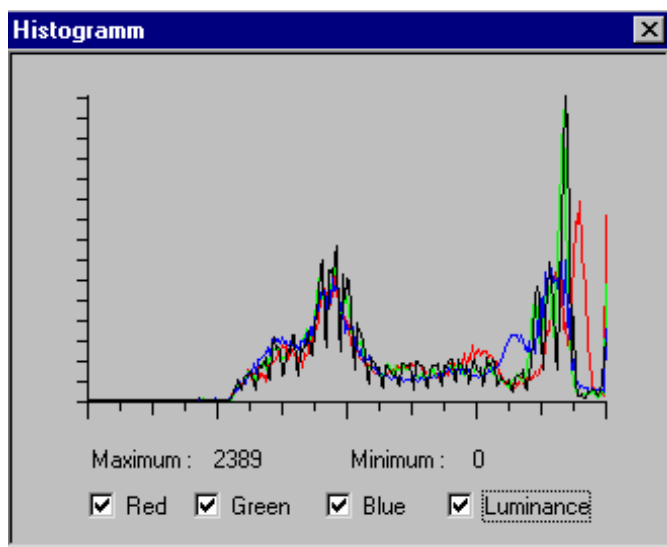


Figure 30: *Histogram*



The X-axis includes the values between 0 and 255. Using the check boxes in the histogram window, the curves of grey values, or the separate color values, can be turned on/off.

**Caution:**

A histogram can only be created from a static image, and not from a live image. To create a histogram for a live image, you must first create a static image using the snapshot function.

- **Analyzing Colors:**

Selecting the *Color Meter* option opens the window shown in Figure 31.

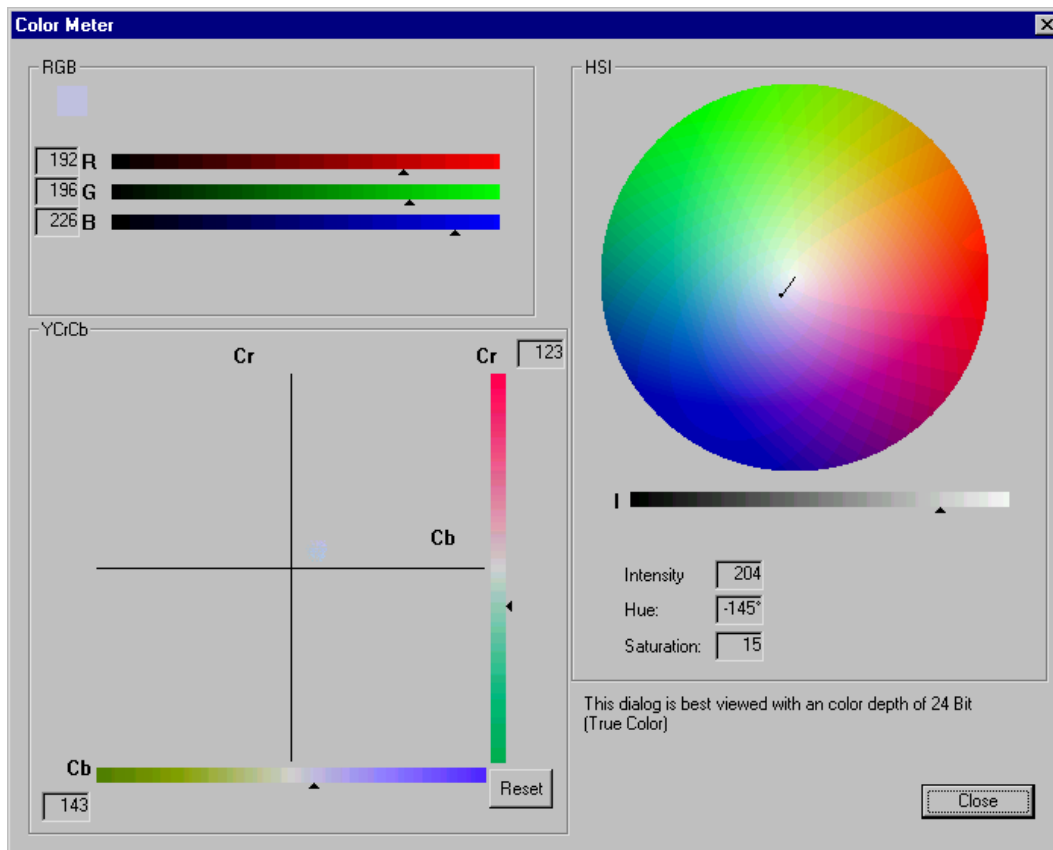


Figure 31: *Color Meter*

The color meter option only functions in the live image display. The color meter displays various color models for the color values of pixels embedded in the center of the image.

A small crosshair that appears in live image indicates the center of the image.

The RGB model displays the color values for red, green and blue as pointers and number values on the intensity bar.

The YCrCb model displays color values as color bars and in a coordinate system.

Thus, color fading and changing can be observed over an extended period of time.

The *Reset* button erases the existing coordinate graph and creates a new graph.

The HSI model displays the color values in a color circle. The length of the vector indicates the saturation level, and the direction of the vector indicates the hue. Brightness is displayed in a gauge at the bottom of the window.

Each value is also numbered.

- **Displaying Color Bars:**

Select the *Color Bars* option in order to test the Grabber.

The color bars are generated from hardware and not the demo program. The number of bars displayed depends on size of the chosen image. All color bars are displayed with a horizontal resolution of approx. 515 pixels.

- **Arithmetic Operations on Static Images:**

The *Arithmetics* menu option provides some simple arithmetic operations on static images (see Figure 32).

For example, images can be added, subtracted, multiplied or divided pixel by pixel. In addition, a constant can be added to each pixel (brightness changes) or the constant can be multiplied with each pixel (contrast change).

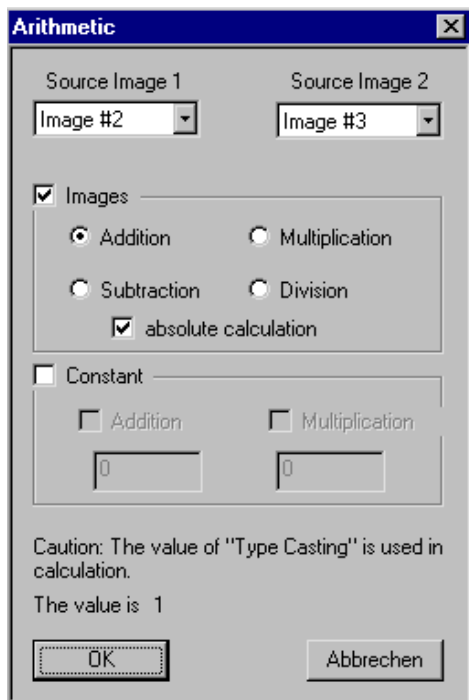


Figure 32: *Arithmetics Menu*

Images to be manipulated can be selected from *Source Image 1* and 2. The number behind *Image#* corresponds to the number of the image window.

Arithmetic operations can be selected from the *Images* entry. The user can also choose to perform an absolute calculation.

When performing an absolute calculation, negative values are not allowed. Eventually these negative values will display a meaningless and incorrect result.

Under the *Constant* option, a constant can be added to each pixel (changes brightness) or can be multiplied with each pixel (changes contrast).

All arithmetic operations are normalized. This is important if the result is expected to be outside of the displayed range of values. (Each color channel has a range from 0 to 255). On principle, values greater than 255 are set to 255, and pixels with values less than 0 are set to 0.

This prevents, for example, the creation of white images caused by multiplying pixels.

The normalization factor can be selected from *Options* pull-down menu under *Type Casting Settings* (see Figure 33).

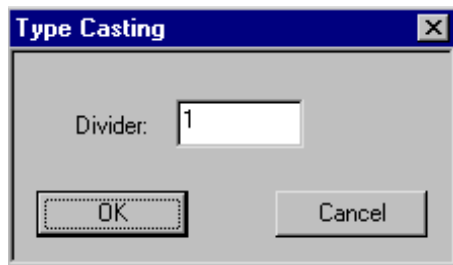


Figure 33: Selecting the Normalization Factor

The actual value is displayed in the bottom section of the *Arithmetic* menu.

**Caution:**

Incorrect settings of the normalization factor will provide unsatisfactory results with arithmetic operations (i.e black or white images).

- The *Add Live Images* option enables up to 1000 consecutive live images to be compiled into a single image.
- The desired number of images can be selected under *Options / Addition Settings* (see Figure 34).

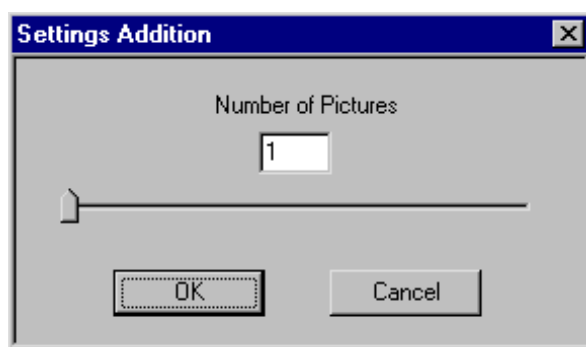


Figure 34: Number of Images

This feature can also be used to reduce noise levels when recording images or to reduce the resolution of moving objects, in comparison to the background.

After the addition process the resulting picture is normalized so that the original brightness is retained.

The length of the process depends on the number of images that were added and the capability of the computer.

The operation's status is displayed as a percentage in the lower section of the window.

**Caution:**

In order to ensure that the brightness for added images has the same quality as single images, the parameters for the number of images change simultaneously with the normalization factor (type casting).

The normalization factor must eventually be re-configured when using additional *Arithmetic* functions.

• **Option Port Test:**

The menu *Features/Test Hardware* contain the subitem *Option Port*.

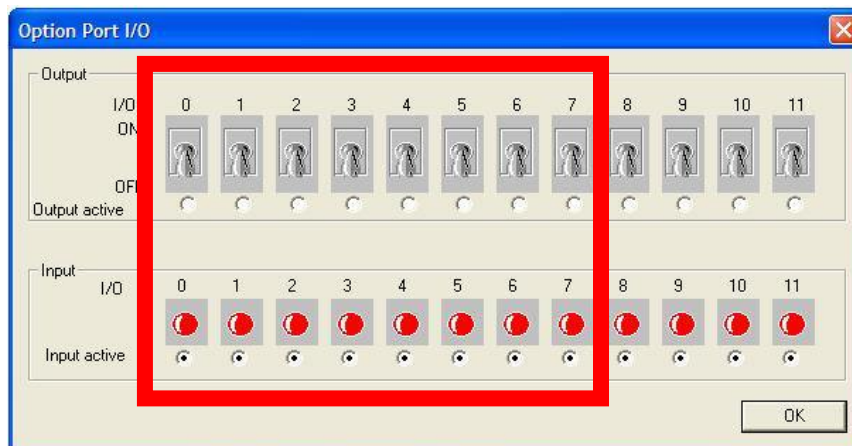


Figure 35: Option Port Menü

This Menu permit to control the I/Os on the Option Port (X300). It is possible to turn the I/Os on or off further it possible to set the I/Os as an input.

- **Jumper reading:**

Der Menüpunkt *Features/Test Hardware* beinhaltet den Unterpunkt Option Port.

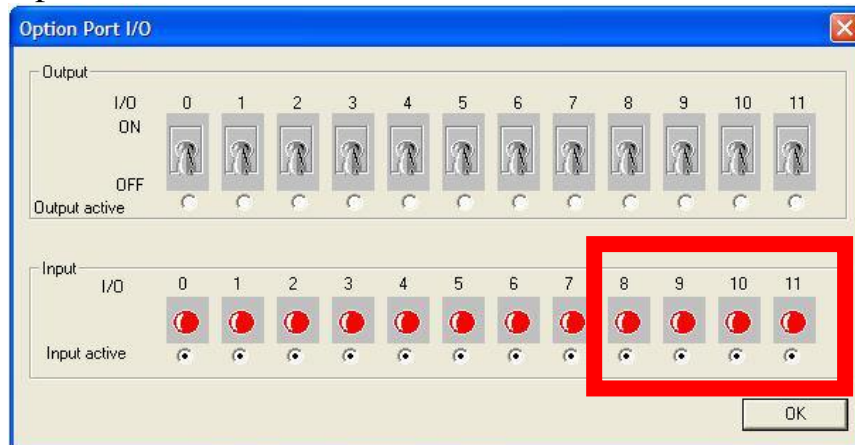


Figure 36: Option Port Menü (Jumper)

In this menu it is possible to read the states of the jumpers. If no jumpers are equipped on I/O 8 to 11, they are on high level. They are red marked. If jumpers are equipped the red selection goes out.

## 5.7 Storing Images, Ending the Program

The menu option *File* enables users to store live images (snap shots), static images and arithmetically processed images. The options *Save* or *Save as* allow the images to be saved with an index number given by the program, or with a name given by the user.

The images are saved in bmp format and can be viewed and processed with any graphic program.

The *Close* option consecutively closes static images as well as the live window.

*Exit* closes and leaves the program.

## 5.8 Getting Started with Linux

For the pciGrabber-4x4 can be used the BTTV-driver. This driver is already included in most Linux distributions. You can work without special driver. At the moment the driver must be still customised by hand. Please, read the FAQs under [www.PHYTEC.de](http://www.PHYTEC.de).

### Hints:

- The use of the PCI Express-to-PCI-Bridge can cause that the PCI bus allocation for the devices not fit, because Linux update these not independently. The devices must be relocated. The graphics card can be configured with the SaX2 function automatically or by hand in Xorg.conf. If further information is needed please ask your Linux provider.



## **Part 2**

# **Programming Manual**

## 6 Driver Software

This section gives you the information how you can access the pciGrabber-4x4 with your own program.

The driver library provides you with a collection of functions, which are able to configure the Grabber, which can inquire the status of the Grabber and start the digitization.

Software drivers for different operating systems are available.

In this manual drivers for

- Windows XP/VISTA
- Windows NT 4.0
- Windows 2000

are explained.

### **Note:**

In order to obtain the newest information regarding the driver and the availability of additional drivers, please read *readme.txt*. (This file can be found on the installation CD.)

The next section describes the technical features of the Grabber and explains television norms in greater detail for a better understanding of the Grabber's functionality.

## 6.1 Technical Basics

### 6.1.1 Block Diagram of the pciGrabber-4x4

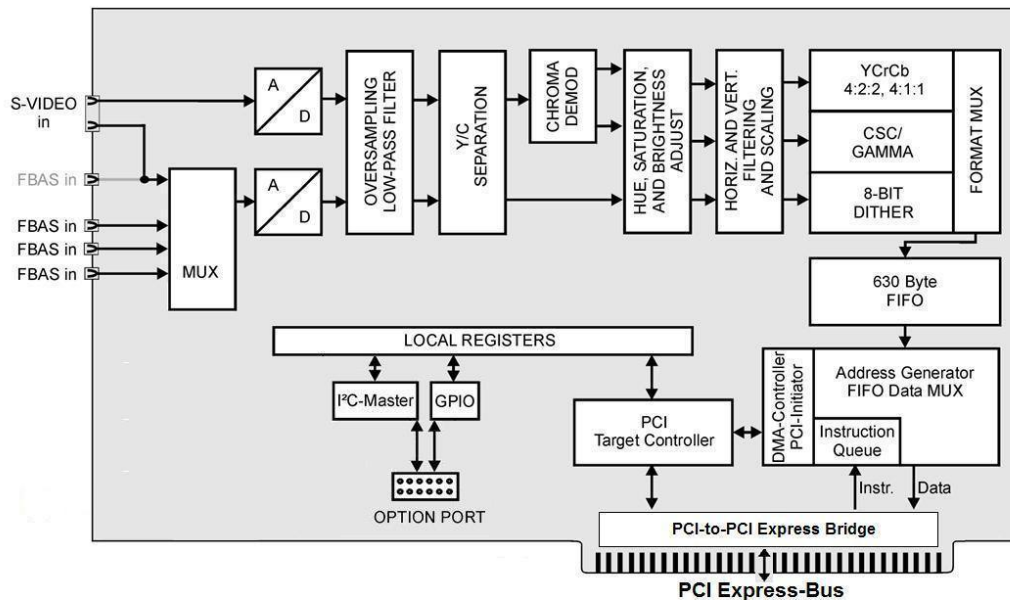


Figure 37: Block diagram VD-012

Figure 37 shows the block diagram of the pciGrabber-4x4. The composite input signal is connected to a 9:1-video multiplexer, which is controlled via the PCI-Bus. The following A/D-converter digitizes this signal. All image sources can be used, which provide a color video signal corresponding to the CCIR- standard „PAL (B,D,G,H,I)“, „NTSC (M)“.

In Germany image sources generally provide PAL-signals. In this manual we assume that always PAL-signal sources are used.

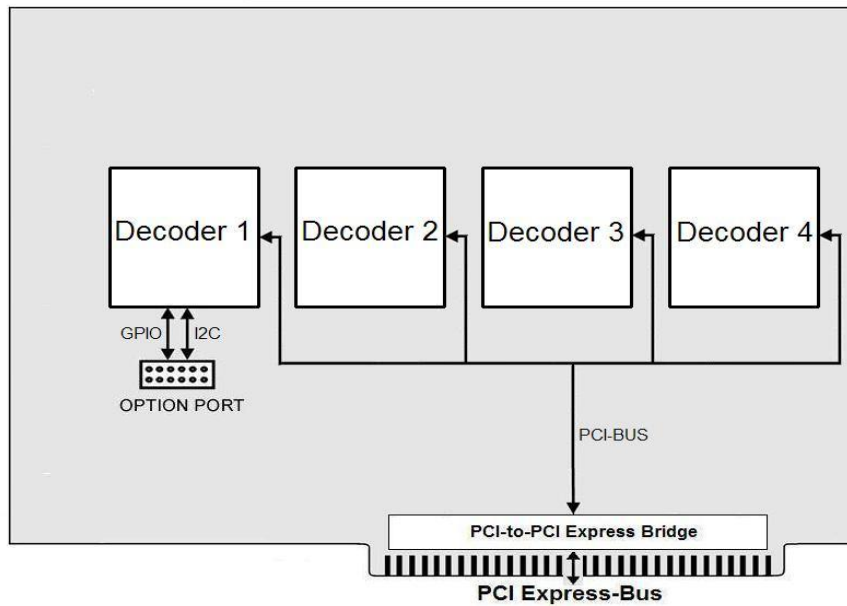


Figure 38: Block diagram VD-012( part 2)

Via the S-VIDEO-input luma- and chroma-signal can be supplied separately (for example from a S-Video-camera or S-VHS-videorecorder). For the spectral component of the color a separate A/D-converter is used, which improves the quality of the image.

Also black/white videosources can be connected to the pciGrabber-4x4. The processing of grey scale pictures with 256 grey graduations is already provided in the Grabber and can be activated by software. Applying black/white sources, the sharpness of the image can be improved by deactivating the luma notch filter by software.

After the A/D-converters follow operational components, which decompose the data stream of the image into its components: After the chroma-demodulator the data are separated according to brightness. (Y) and color portion (Cr/Cb). Subsequently follows the digital correction of brightness, contrast, color saturation and the size and resolution of the image.

The following *video format converter* produces the data formats, which are provided by the pciGrabber-4x4. Via a datamultiplexer the required format is selected and stored in the 630 Byte FIFO memory. The FIFO is an interface to the following PCI-bus interface, which is responsible for the data transfer through the PCI-bus.

The PCI Express-to-PCI-Bridge changes the PCI bus in an PCI Express bus.

The image data are transferred by DMA to the main memory of the PC. For each field a separate DMA-channel is used. The transfer can be organized in different ways. For this reason a *pixel instruction list* for each field is used, which is denoted *RISC-Program*, for the PCI-controller of the pciGrabber-4x4.

Via the PCI Express-controller the access to the *local registers* is managed. This allows the adjustment of the parameters of the Grabber and the acknowledgement of the actual status. This registers are also used to actuate the I/O- lines defined by the user, and to drive the I<sup>2</sup>C-interface integrated in the Grabber.

### 6.1.2 The Videosignal and Digitization

The standard videosignal, which is processed by the Grabber, contains 625 lines, which are divided into two fields (see Figure 39). The first field (odd field) contains lines 1 to 313, the second (even field) the lines 314 to 625. The fields are interlaced, in order to reduce flicker of the TV-picture. In special respect line 314 follows after line 1. Besides various retrace- and blanking lines, the videosignal contains lines for control and data purposes and lines for videotext information, which restricts the actual image size to two fields of 288 lines.

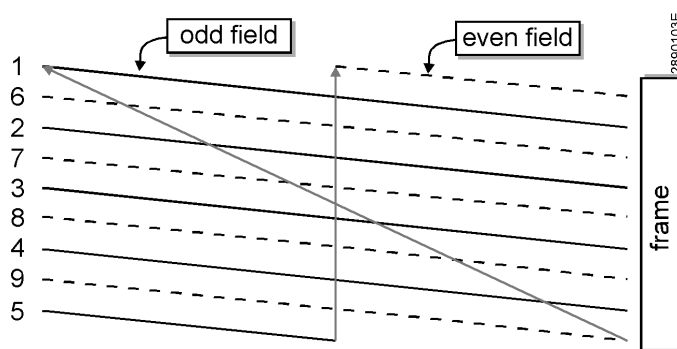


Figure 39: *InterlacedImage (Example with 9 Lines)*

Each field is built up within 20msec. One field provides already the whole image, but the vertical resolution is reduced to the half. For many applications this might be sufficient, so that after 20 msec a digitized image is already available. In case of that the resolution in X-direction can also be reduced, we can obtain an image without distortion. However a reduction of the resolution in X-direction can not speed up the digitization process, since the time base is fixed.

If the full TV-resolution is required, time has to elapse until both fields are digitized (40 msec). Both fields follow one after another.

In order to make the interlacing of both fields possible, the last line of the odd (the first) field, is reduced to the half. Therefore the first line of the second field contains only the half line.

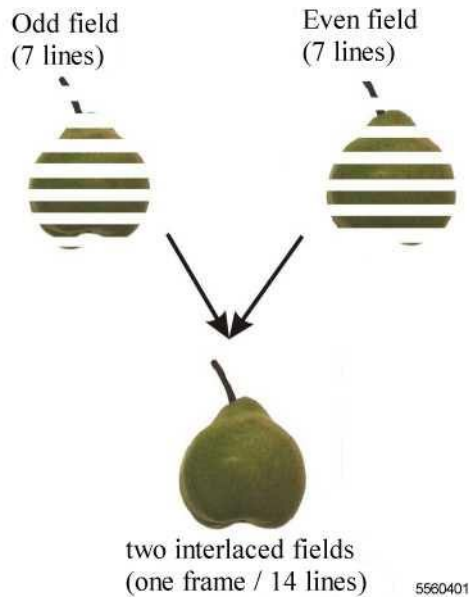


Figure 40: Fields and Frames

For fast moving objects it might happen that the time between the digitization of the first and second field is so long that meanwhile the objects have moved some distance and both fields don't match anymore, which will cause some remarkable blurring. For this reason quite often only one field is used with a reduced resolution.



Figure 41: Moving Objects Cause Comb Effects

### 6.1.3 Transfer and storage of color

Color and brightness are always separated for the transmission by the TV-systems. Transmitted are the brightness (luma signal, Y-signal) and the color differential signal (chroma signal). This signal defines the color of a pixel by the color tone and color saturation.

The TV-standards reduce the bandwidth of the color signal in comparison to the brightness signal. The color of a pixel is more 'blurred' than its brightness. This corresponds to a painter, who at first makes a sketch with a sharp pencil and then colors the areas with a broad brush.

The Y-bandwidth of the PAL (B,G,H,I) - system is 5 MHz, and the bandwidth of the chroma signal is 1.5 MHz.

The chroma signal is also denoted as U/V signal for PAL standard or Q/I-signal for the NTSC standard. V- and I-signal define the reddish colors, whereas the U- and Q-signal define the bluish-violet colors. Altogether we speak from the Cr/Cb signal (chroma red/ chroma blue).

With the triplet (Y,Cr,Cb) the brightness and color of a pixel are completely defined. These values are ready to be used without further evaluation for image processing in respect to color recognition or color control.

Frequently the definition of a pixel is preferred in the red, green and blue (R,G,B) notation. The transformation according to CCIR-recommendation for PAL is achieved with the following matrix:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1,371 & -191,45 \\ 1 & -0,338 & -0,698 & 116,56 \\ 1 & 1,732 & 0 & -237,75 \end{pmatrix} \cdot \begin{pmatrix} Y \\ U \\ V \\ 1 \end{pmatrix}$$



The pciGrabber-4x4 is able to convert the images into the RGB-format and stores the RGB- color triplets in the memory. This format provides a good base for further processing.

Often the YCrCb-format is more suitable for storage or transfer of image data, since the data volume is less. Instead of three Byte only two Byte (one word) are required. The lower eight bits contain the brightness and the eight upper bits specify the color portion (Cr/Cb).

For each Y-value alternatingly only one color portion Cr or Cb is associated. So each pixel has only one part of color information either the red or blue portion. The missing information can be obtained from the neighboring pixel. The color is transferred and stored only at the half resolution of the brightness. Since the bandwidth of the color information is already reduced by the TV system, this procedure does not mean a real restriction. This data format is denoted as YCrCb4:2:2.

The first pixel of each line delivers Y1,Cr1/2, the second Y2,Cb1/2 etc.

For the correct recognition of the color information of an image, four subsequent fields are required to be digitized. Therefore it is not sufficient to connect the video source only for a short period or to connect the videosource only for the duration of the digitization of one field.

In addition the recognition of a field might not work correctly at the beginning, in case another not synchronized signal from a camera is applied. In case of fast switching between two signal sources the digitized image might be incorrect and it is recommended to observe some time delay.

#### 6.1.4 Data storage by DMA and RISC-Program

This section describes the transfer of the data to the main memory and the storage of the pixels to the addresses specified by the user.

As mentioned before, the transfer of the data is accomplished via two DMA-channels, one for the odd and one for the even field. During the time of digitization the DMA-controller of the pciGrabber-4x4 is controlling the PCI-Bus and is *master*. The data are transferred in real time along the PCI-bus to the main memory. This is possible because of the high transfer rate of the PCI-bus.

The picture data are sent in real-time over the PCI bus to the PCI Express-to-PCI-Bridge. From there the data are sent over the PCI Express bus and will transmit into the main memory. This is possible by the high transmission speed of the PCI and PCI Express bus.

Delays of the data transfer or for time intervals the PCI-bus is not available to the Grabber (that means some other devices become master), are bypassed by a FIFO-memory. This allows only a short time span to bypass the blocked bus, since otherwise an overflow might occur and portions of the image are lost. The bus is controlled by the parameter *Maximum\_Latency* and *Minimum\_Grant* of the PCI-board. If required, this parameters have to be adapted to the data transfer rate, to the system configuration and to the bus performance.

The pciGrabber-4x4 is very flexible concerning the storage of the data. The user can specify destination and format of the data within a certain scope. For this a mechanism is required to separate the continuous flow of data into partitions and direct the data to the required addresses.

This mechanism is accomplished by the pciGrabber-4x4 with the help of the *pixel instruction list*. This is a RISC-program, which drives the DMA-controller correspondingly.

This RISC-program has to be written by the user and must fulfill the required tasks and has to match the data and image format. So the

program has to be specified according to each problem, which implies that the RISC-program is created during run time of the user program, since often the parameters (for example the size of the image) which control the RISC-program, are variable or not available prior to this time.

The software driver delivered with the pciGrabber-4x4, creates the appropriate RISC-program automatically with the adjustment of the image size. This process is transparent to the user program.

Nevertheless this feature should be in the conscious of the programmer using this driver software.

Figure 42 depicts this scheme. For image size and data format selection the user program applies the function *set\_image()* of the driver. The driver starts two actions: first the image size is set in the *VideoScaler* by values in the local registers of the Grabber via the PCI-bus. This implies, that the pciGrabber-4x4 creates the correct image size and the data flow has the correct format and provides the appropriate synchrony signals. In the same way the *DataFormatConverter* is adjusted to the correct format. This implies that the flow of pixel data to the FIFO has the correct format (for example RGB).

The second action of the driver software is the creation of a data flow appropriate to the RISC-program, which is stored in the main memory of the PC. The DMA-controller of the pciGrabber-4x4 is notified of the starting address of this program. During digitization, the DMA-controller receives the RISC-commands in sequence by DMA from the main memory and executes those commands and stores the data according to those instructions.

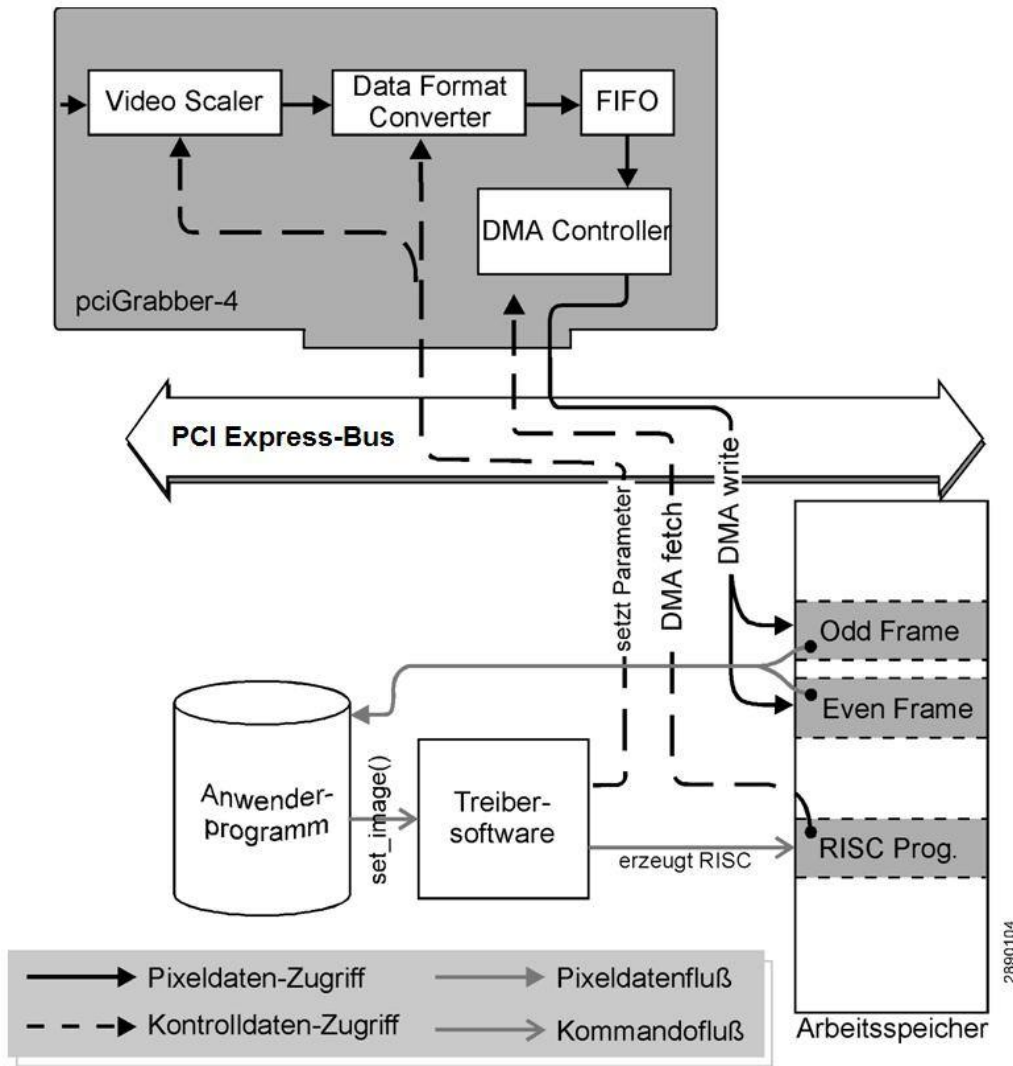


Figure 42: Pixel- and Control Data Flow (Overview)

The flow of data via DMA-access is directed to the main memory to the address specified by the RISC-program. This address region is reserved by the user program (e.g. the definition of arrays).

The regions might be defined - as shown in Figure 42 - as two separate regions, one for the odd- and one for the even field, or only one region, which is provided for the whole frame of the pciGrabber-4x4. The different options are selected by a parameter in the `set_image()` function, which influences the creation of the RISC program via the driver.

Since the user program defines the address regions it knows the position of the memory regions and might mark them by a pointer. In this way the access to the data can be accomplished without using the driver.

The driver provides information of the status, which indicates the end of the storage of the image in the memory, so that at this time all data are available.

This mechanism guarantees a fast access to the data. The process is real time regarding to the standard TV format. For the digitizing of a field it takes a time of 20 msec and the digitizing of a whole frame lasts 40msec. In some cases a time delay must be added to get the total time from the demand of the image to the end of the digitization process.

This additional time might arise from waiting for the appropriate field. For example, if an even field is demanded, but the camera has just started scanning an even field, so it is necessary to wait until this field and the next following odd field are finished. In the worst case a delay of 40msec (two fields) can be expected. Now the demanded field can be digitized, which will last another 20msec. During the following 20 msec nothing will happen in this memory region since the odd field will be received. The next 20msec a new even field is stored in the memory. It must be considered that the old information will be overwritten by the new information, otherwise the content might be interpreted incorrectly by the software especially for moving scenes.

Now we consider the case of digitizing a whole frame in one memory region. Here the same effect might occur but in some different fashion. After 20 msec digitization the information for an odd field is completely available and therefore all odd lines are defined, but the even lines are not defined. During the following 20 msec the data for the even field are received. Therefore there is no time, which is not used to transfer data to the memory except for the blanking interval. So there is always a point (X,Y) where old and new information are stored adjacently, so that a mismatch will show up.

## 6.2 Driver for Microsoft Windows

When executing the installation program for the Windows demo program, the files are downloaded and stored to the hard disk. The structure of the file directory is similar to Figure 43. The window on the left-hand side displays path names. These path names can be edited during installation in order to create user specific names for the system. The libraries and *include* files to be compiled are located in the labeled subdirectories.

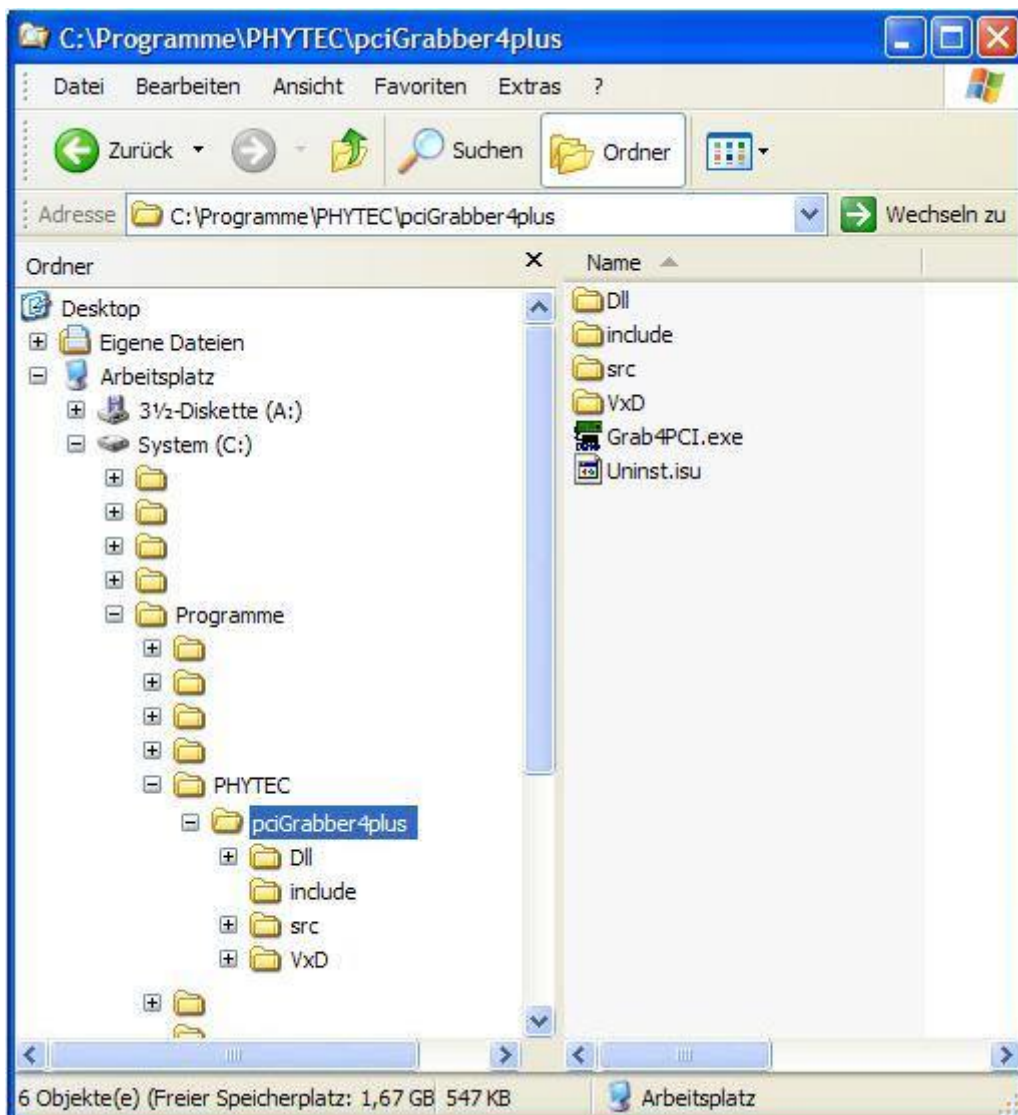


Figure 43: Directory for Window's Driver

### **6.2.1 Requirements**

Programs for the pciGrabber-4x4 can be created with the help of various development environments.

These newly created applications will work with the Windows XP/VISTA™ and the Windows NT4.0/2000 operating systems.

**Caution:**

The device driver and corresponding DLLs must be copied into the Windows main directory in order to implement the pciGrabber-4x4 in a Window's operating system. In addition, the system driver must be registered into the registry table.

Phytec's installation program automatically copies the device driver and DLLs and registers the system driver. Therefore, all the requirements for operation are fulfilled.

Creating corresponding installation routines is recommended when installing newly created applications onto other computers.

### 6.2.2 Application of the Device Driver for Windows NT4.0

Because the pciGrabber-4x4 accesses by DMA access to the memory, it is necessary that it concerns physically attached and continuously addressable memory. Because of that the physically memories which acts as an image storage must allocate. Moreover, the driver permits the access to the registers pciGrabber-4x4.

In order to implement Windows NT4.0 with user specific applications, use the included installation disk from Phytec.

These applications can be found on the installation CD, in the **PCIGRAB4\DRIVER\WINNT40** directory. The files stored in this directory can be copied to a disk and run with user applications.

The user also has the option to create an installation program specific to user needs. When creating this program, please take note of the following points:

The driver must enter the Windows NT4.0 system into the register. This can be done in the following manner:

- The *PCIGRABBER4.SYS* file must be copied into the directory labeled *<Windows>\System32\drivers*.
- The driver is then entered into the *Registry Table*:  
Use the REGEDIT program (located in the WindowsNT directory).  
Scroll down the directory tree  
*HKEY\_LOCAL\_MACHINE/System/CurrentControlSet* and select *Services* (see Figure 44).



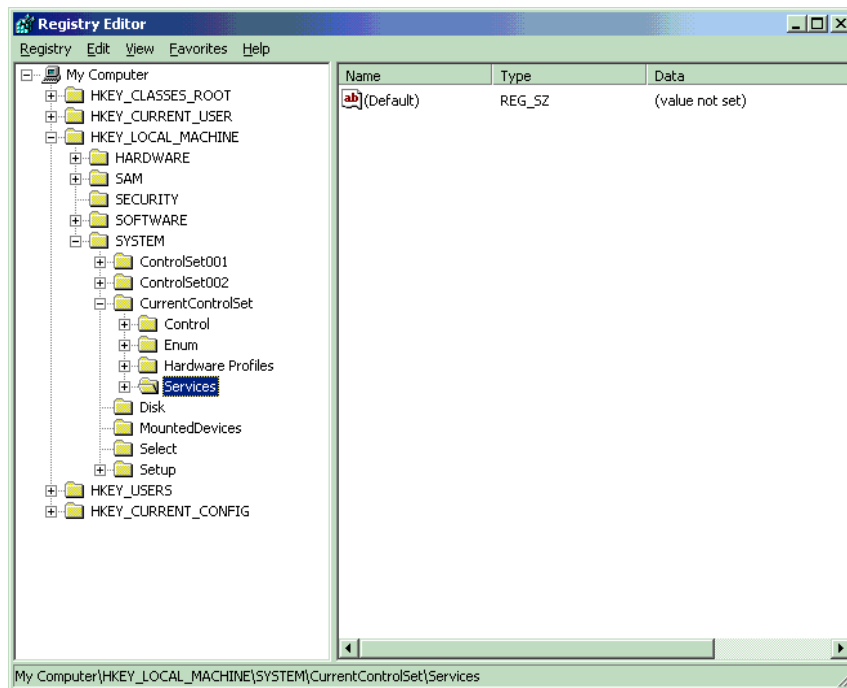


Figure 44: Windows NT Registration Editor

Open the *Services* folder. Select the *Edit/New/Key* pull-down menu and a new key will be created. Name this new key „pciGrabber4“, as shown in Figure 45.

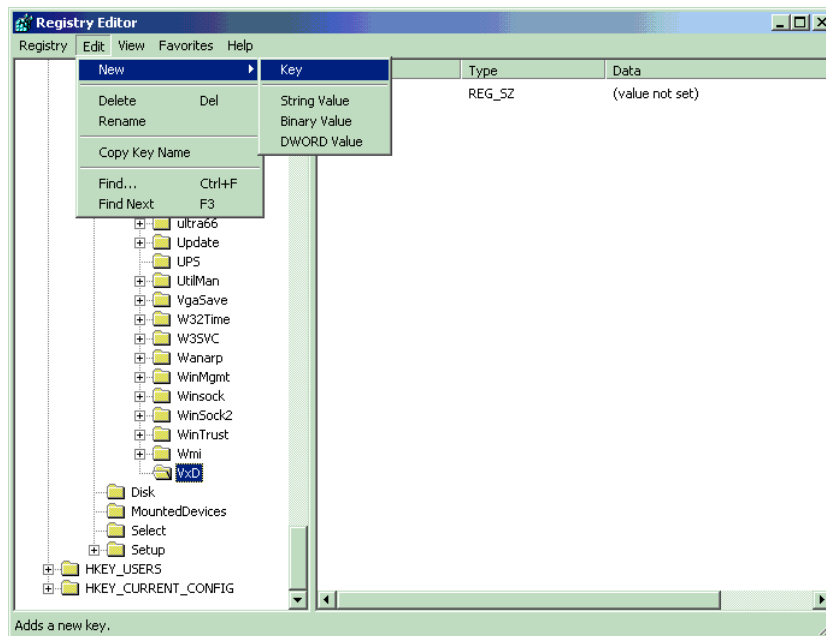


Figure 45: Entering a Device Driver

Now configure the new key group and mark the entry with „pciGrabber4“ as shown in Figure 46.

Select the *DWORD value* command from the *Edit/New* pull-down menu. A new entry named „New Value #1“ will be created within the „pciGrabber“ key.

Change this name in „Start“. Right click on the newly created entry and select *Modify*. In the dialog box that will appear, enter the number 2 into the *Value* field. Select the *DWORD value* command option from the *Edit/New* pull-down menu. Change the description in „Type“ and enter a value of 1.

Similar to the previous DWORD entry, select *DWORD value* from the *Edit/New* pull-down menu and enter a value of 1 for „ErrorControl“. The end result should look similar to Figure 46.

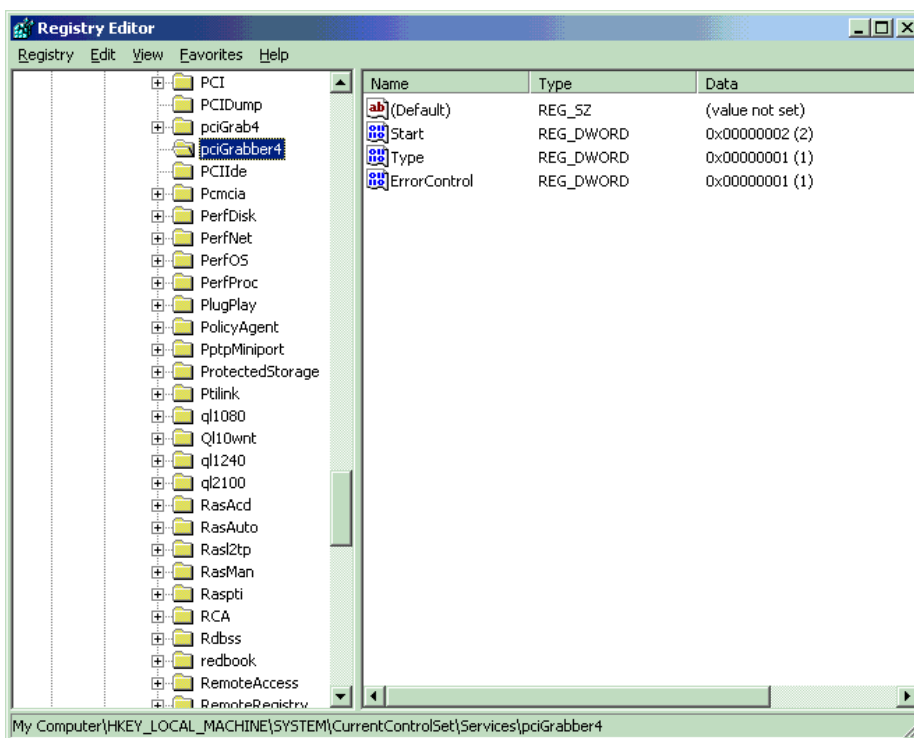


Figure 46: Configuring the Driver

After restarting the computer, the driver is automatically loaded when starting Windows NT.

**Caution:**

The device driver must also be removed when uninstalling user programs. Erase the entry from the Registry Table and from the system directory.

The driver reserves 1.2 MB for the pciGrabber-4x4 in the main memory. The memory space is not available for other applications.

**Caution:**

Pay careful attention when changing the registry entries. If an error occurs while making these changes, the configurations can be permanently damaged. This could render the Windows NT operating system inoperable.

For simplicity, it is recommended that end users implement an install/uninstall program, since all of these tasks are automated.

### **6.2.3 Application of the Device Driver for Windows 2000 / XP / VISTA**

Similar to Windows NT4.0, the device driver allocates physical memory for storing images. The driver also allows access to the Grabber's register.

Reserving memory space is only possible with a device driver under the Windows 2000 operating systems.

The driver also transforms linear memory addresses into physical memory addresses.

User programs do not have direct communication with the driver, instead access is provided by DLLs.

Phytec's installation disk is recommended for operation of the Windows 2000/XP/VISTA driver with user specific applications.

These applications are located on the installation CD under the ***PCIGRAB4DRIVER\WIN2k\_98*** directory. Files from this directory can be copied to a disk and applied to user applications.

## 6.2.4 Application of the DLL

The DLL provides communication between user programs and the pciGrabber-4x4. The DLL configures the Grabber and controls digitization events. In addition, the DLL allows access to the data of digitized images stored in the main memory.

**Caution:**

The DLL is not linked to the user program, but invoked during runtime. Therefore, the DLL must be available in the Windows system directory during program runtime.

In addition to GR4CDLL.DLL the following DLLs are necessary for operation:

- **MSVCRT.DLL**
- **CTL3D32.DLL**
- **MFC42.DLL**

Windows provides various API functions to dynamically link the DLL. **Load Library(...)** is used to load the DLL and a handle is subsequently returned for the DLL. The API function **GetProcAddress(...)** provides starting addresses for various DLL functions. In order to release DLLs at program end, call the function **FreeLibrary(...)**. For more information, please *refer to the development environment's User's Manual/Data Sheets or refer to the enclosed SDK source.*

### 6.2.5 Application of the Windows XP/VISTA™ Windows NT4.0™ / Windows 2000™ DLLs

In order to use the DLL *Gr4CDLL.DLL*, the software developer must define a function pointer for each function that will be used in the application.

#### Example:

- Function to be used: **WORD Get\_Error(void)**
- Definition of the function pointer:

```
WORD (PASCAL * lpfn_GetError)(void);
```

Use **GetProcAddress(...)** to obtain the relationship between the function pointer and the DLL.

#### Example:

```
lpfn_GetError = (WORD(PASCAL *)(void))  
                GetProcAddress(handle, „Get_Error“);
```

The function can now be called with:

```
WORD Errorstatus;  
...  
Errorstatus = lpfn_GetError();
```

#### Caution:

Check the value of the function pointer (return value from GetProcAddress) to be sure that it = 0. A value of 0 ensures that the driver version installed on the user's computer supports the functions and will return a valid handle.

## 6.2.6 Programming under Delphi

In order to utilize the pciGrabber-4x4 with Borland Delphi for user applications, the driver-DLL with function export "Gr4CDLL.DLL" should be employed.

To introduce the functions to the DLL in Delphi a corresponding *Unit* has to be defined. Please pay attention to the correct calling sequence, to guarantee the compatibility of the DLL. Define the functions with the type `stdcall`. In case of false declarations stack-overflows or -underflows can occur, which will cause a violation against protected areas. In the following example a unit is defined:

```
unit grab4dll;

interface

{ The calling sequence 'stdcall' defines the sequence of the pa-
parameter transfer to the stack and signals to Delphi that the
called function frees the stack region, which was used for the pa-
rameter }

function Grab4_Get_Error: word;
stdcall; external 'gr4cdll.dll' name 'Get_Error';

function Grab4_Max_Device_Number: word;
stdcall; external 'gr4cdll.dll' name
'Max_Device_Number';

function Grab4_Data_Present(nDevNo: word): word;
stdcall; external 'gr4cdll.dll' name 'Data_Present';

function Grab4_GetPictureBufferAddress(nDevNo: word;
dwBitsSize: Cardinal): cardinal;
stdcall; external 'gr4cdll.dll' name 'Data_Present';

procedure Grab4_Initialize(nDevNo: word);
stdcall; external 'gr4cdll.dll' name 'Initialize';

procedure Grab4_Set_Channel(nDevNo, nChannel: word);
stdcall; external 'gr4cdll.dll' name 'Set_Channel';

procedure Grab4_Start_Grabber(nDevNo: word);
stdcall; external 'gr4cdll.dll' name 'Start_Grabber';

procedure Grab4_Stop_Grabber(nDevNo: word);
stdcall; external 'gr4cdll.dll' name 'Stop_Grabber';
```

```
procedure Grab4_Set_Image(nDevNo: word;
                          nOhpos, nOvpos, nOhsize, nOvsize,
                          nOppl, nOlines, nOColformat :word;
                          nEhpos, nEvpos, nEhsize, nEvsize,
                          nEppl, nElines, nEColformat :word;
                          nColsystem:word;
                          nInterlaced:word;
                          nSingleShot:word);
  stdcall; external 'gr4cdll.dll' name 'Set_Image';
```

```
const
  NTSC_M:    word    = 0;
  PAL_BDGHI: word    = 1;
  SECAM:     word    = 2;
  PAL_M:     word    = 3;
  PAL_N:     word    = 4;
  AUTO:      word    = 5;

  RGB32:     word    = 0;
  RGB24:     word    = 1;
  RGB16:     word    = 2;
  RGB15:     word    = 3;
  YUY2:      word    = 4;
  BtYUV:     word    = 5;
  Y8:        word    = 6;
  RGB8:      word    = 7;
```

```
implementation
```

```
{ DLL Functions }
```

```
end.
```



### 6.2.7 Description of the DLL in Existing Functions

The user can control all of the events in the pciGrabber-4x4 using the functions of the DLL. The DLL can also read the actual status, as well as configured values. These functions are described in more detail further on in this manual.

The functions have been divided into five groups for easier discussion. The group number is shown in a black circle.

The functions are classified as follows:

**❶ Routines for Initialization / Calling up Hardware**

*This group includes routines that must be called one time prior to using the Grabber, to ensure that the Grabber functions properly. Also included are two functions that give information about the installed hardware and its capabilities.*

**❷ Routines that configure the Grabber configures for the grabbing process:**

*Functions from this group configure the Grabber to the connected image source (camera). These functions also determine the appearance of the grabbed picture as an end result in memory (image size, color format, etc) The user should determine whether each function is needed, and which parameters are necessary. These functions may be called-up several times during the processing of a program (i.e. when the input channel should be switched or if the image size should be changed).*

**❸ Routines for Executing and Controlling the Grabbing Process**

*This function starts the image digitization, monitors the Grabbing process and ends digitization.*

**❹ Routines for Configuring Image Parameters**

*Functions from this group enable configuration of parameters, such as brightness, contrast, saturation, etc. These functions are not necessary, but can be called at any time to adapt the final image to user needs.*

⑤ Routines for Controlling the Option Port

*This category includes functions that do not directly influence the grabbing process, but rather deal with the features of the Grabber, i.e. I/O port, I<sup>2</sup>C interface, etc. These functions need only to be called when a corresponding Grabber feature is implemented.*

**Important:**

In all the following routines the parameter `nDevNo` is used. This parameter identifies the desired pciGrabber-4x4 rather decoder. The number of the installed pciGrabber-4/decoder can be determined by the function `Max_Device_Number()`.

## Compatibility to the pciGrabber-4

The driver is basically downwards compatible. Programs for the pciGrabber-4, pciGrabber-4*plus* and pciGrabber-4 *express* works also with the pciGrabber-4x4.

But pay attention, the forerunner work with an different kind of Option Port and the numbers and mapping of video inputs are different.

Functions that are not compatible with the older driver version for the pciGrabber-4 are denoted with a star (☆).

Please take note of the functions with a ☆ when adding new features from the pciGrabber-4x4 to existing applications.

Most functions are compatible with the pciGrabber-4, although some functions may not be used due to non-compliance with hardware requirements. In any case, the new driver version should be used with new applications.

## ● Evaluation of the Error Messages

### **WORD Get\_Error(void);**

Returnvalue:    0 = no error  
                  1 = device number not found  
                  2 = bad register number  
                  3 = initialization failed  
                  4 = Grabber not found  
                  5 = unknown parameter value  
                  6 = not supported  
                  7 = newer driver version required (update)  
                  8 = no PHYTEC grabber card found  
                  9 = no acknowledge  
                  10 = invalid address  
                  11 = write access denied

Each execution of a driver function should be checked if it was successful. For this purpose there is the function *Get\_Error*. Immediately after the execution of the function, the internal error variable of the driver is set to the actual status.

This variable is available to the user program via the function *Get\_Error*, so that a reaction to this error message can occur.

The investigation of the error variable is possible until a new execution of a driver function has occurred. Then the error status of the new function call is set.

## ❶ Obtaining the Version Number of the DLL

### **DWORD GetVersionNumber (void);**

Return value:      Version number for the Grab4CDLL  
                         HighWord: Major\_Version\_Number  
                         LowWord: Minor\_Version\_Number

The version number for the Grabber-DLL can be obtained using these return values.

#### **Note:**

Test the version number and ensure that when using the pciGrabber-4x4 the Major\_Version\_Number is larger than or equal to 4.

⇒ **❶ Determination of the number of available  
pciGrabber/Decoder**

**WORD Max\_Device\_Number (void);**

Returnvalue: Number of pciGrabber found

This functions specifies the number of pciGrabber-4x4 in the system, more precisely the numbers of decoders in one system.

- VD-012            four decoder
- VD-012-X1       four decoder
- VD-012-X2       two decoder

This is required, because PCI-/PCI Express-devices are not configured with jumpers or other hardware, but are assigned automatically an addressing region, by the PCI-BIOS. With the help of the PCI-BIOS the address region can be determined for each Grabber card. If several cards are installed in the system, the addresses are returned in a sequence (see also section 2.4).

The user has not to care about addresses or address regions when using the driver. Those are converted internally into *device numbers* (= nDevNo). Each pciGrabber-4x4 card in the system is assigned a unique device number during generation of the Grabber class. It can not be predicted for certain which number is assigned to which card, since this depends on the topology of the bus and the function of the BIOS.

In order to drive the different pciGrabber-4x4 independently by the software, the device number is transferred as parameter with each function call.

The function `Max_Device_Number()` is applied to find out how many pciGrabber-4x4 are installed in the computer. The highest permissible device number is returned. At the same time this is the number of grabbers in the system, since the lowest device number =1.

If the returned value is 0, the PCI-BIOS did not find a pciGrabber-4x4.

For nDevNo only values between

`0 < nDevNo <= Max_Device_Number ( )`

are accepted.

## ❶ Initialization of the Grabber and driver after activation

**void Initialize(WORD nDevNo);**

This routine initializes the Grabber and the driver software after turning on the system. This routine **must** be called before the first access to a decoder. That is for the variant VD-012 and VD-012-X1 the function must be called four times. For the VD-012-X2 two times. This initialization must be carried out for each separate Grabber to be used. This means, that *Initialize* has to be called for all permissible values of nDevNo.

## ❶ Reading Information on the Installed Grabber

**short Read\_GrabberInfo(WORD nDevNo, WORD wInfoType)**

wInfoType: specifies which Grabber feature will be called

return value: Value of the specified feature

This functions delivers information on the hardware, in order to ensure optimal adaptation of the applications to the Grabber.

This function can also be used to define the number of input channels that the Grabber will support and the channel selection dialog can conduct a field test.

The properties are returned with numerical values (from type WORD). The key number is described in the entries of the Header file. The header file is located after the installation of the demo program in the subdirectory "include" of the chosen directory of the demo program. The header is called Grab4DLL.H.

wInfoType can be used to select which information will be called. A description of the parameters is also included in the Header file. If a parameter is not defined, then the function returns a value of 1. An error status has a value of 6 = „NOT\_SUPPORTED“.



The following parameters can be called:

**GRABBER\_TYPE:** Specifies the type number of the Grabber. The return value is numeric, and the Header file includes a description. For example: `Read_GrabberInfo (1, GRABBERTYPE) = 11.` Therefor, Grabber number 11 is VD-012 Decoder 1.

**MAX\_CHANNEL :** Returns the number of composite input channels. For example, for VD-012 = 4.

**Note:**

The call-up features can be expanded with newer card versions. Any available information can be found in the Header file.

## ❶ Read Grabber Name as a Text String

**WORD Read\_OrderCode (WORD nDevNo,  
                          unsigned char\* sCodeString,  
                          DWORD dwSizeOfString)**

**\*sCodeString:** Pointer points to a declared string (25 Bytes min). The function writes the Grabber name into this string.

**dwSizeOfString:** Size of the reserved array

**return value:** Error Code

This function allows the Grabber's description to be read in clear text. A string denoted with zero transmits the name. In order for the name to be transmitted, a character array must be reserved and a pointer must be handed over to the array in `*sCodeString`. The available size of the array is given by the parameter `SizeOfString`.

The size should be at least 25 characters.

If the type description does not fit in the reserved array, the function returns the value 5, an error code.

The type description corresponds to the order number. If a card does not have any clear text information available for the driver, the string „TYPE CODE=xx“ is returned. Xx = encoded type number (see *Read\_GrabberInfo*).

**Note:**

If the pciGrabber-4 (VD-007) or a related product is called up, then the error code 6 and the string „VD-007 or compatible“ is returned.

## ② Grabber setting to the color-system

**void Set\_Color\_System(WORD nDevNo, WORD nColSys);**

nColSys: Code for color system

With the function **Set\_Color\_System** the Grabber is configured for the color system used. Clock frequency and input registers of the video processor are set accordingly. The user can select for *nColSys* predefined constants:

- PAL\_BDGH1 configures the Grabber for the application of PAL-video sources.
- NTSC\_M configures the Grabber for NTSC-sources

---

## ↔ ② Recognition of the video format

**WORD Get\_Video\_Status(WORD nDevNo);**

return value: 0 = 525 line format (NTSC / PAL-M)  
1 = 625 line format (PAL / SECAM)

This function provides the recognition of the video format of the camera at the selected channel, and distinguishes if the source is a NTSC- or PAL/SECAM-system. The distinguishing feature is the different number of lines for both TV-systems. For the recognition, it takes 32 consecute fields from applying the image source, until the identification is finished.

## ↔ ② Configuring the Composite Mode (Composite Inputs)

**void Set\_Composite(WORD nDevNo);**

Calling this routine switches the Grabber onto the composite mode. The chroma ADC is subsequently switched off and the luma notch filter is activated. This mode must be configured when the composite signals are to be digitized.

The composite mode must be selected for all standard cameras (black and white, or color). These cameras do not have S-Video outputs to connect them to the Grabber. Composite cameras are connected to the Grabber via, i.e. the WK-037 and the WK-039 cables. The composite mode is configured during standard initialization.

### **Note:**

*Set\_Channel* must be told in which channel the image will be digitized after calling *Set\_Composite*.

Input:     MINIDIN     = Mini DIN socket is the input  
              AUTO        = automatic configuration

NO\_SIGNAL = signal not found (AUTO mode)

S-Video sources are color cameras that have a special output in order to separate brightness and color signals. These cameras can be connected to a Grabber with either the WK051 cable.

An S-Video source can either be connected to the Mini DIN socket or to the lower HD-DB-15 socket. **The S-Video source can not be connected to both sockets simultaneously!**

© PHYTEC Messtechnik GmbH 2008 L-720e 0

If a signal is found, the Grabber is configured to the Mini Din socket and the parameter `MINIDIN` is returned. If an active signal has not been found, then the Grabber tests the S-Video connector on the Combi socket (second HD=DN-15 socket). If an active signal is found on the Combi socket, then the Grabber is configured to the combi socket and the parameter `COMBI` is returned.

If a signal has not been found at either of the sockets, then the Grabber is configured to the Mini DIN socket and the return value is `NO_SIGNAL`.

**Note:**

- The `AUTO` function does not work when the connected video source does not supply a video signal.  
This function is not compatible with older driver versions.

## ② ☆ Configuring the Input Channels

**void Set\_ChannelEx(WORD nDevNo, WORD nChannel);**

nChannel: Input channel to be configured (1...4)

This function is used to select the input channel. The signal is directed by a multiplexer. The multiplexer is integrated into the video decoder.

Values:

VD-012:	Channel numbers 1-9 allowed
VD-012-X1:	Channel numbers 1-3 allowed
VD-012-X2:	Channel numbers 1-3 allowed

### **Note:**

Configuring the input channels is not necessary when using S-Video sources! The function *Set\_S\_VideoEx()* switches the channels automatically.

### **Caution:**

For the following reasons, when switching input channels, stop inhibit times are to be taken care of until an image for the new channel has been digitized,.

- Definition of the video signal normally lasts up to 4 fields before synchronization is implemented and the color system functions correctly.
- Due to DC voltage decoupling between the Grabber and the camera, different average DC levels on signal lines are present. This can cause charge transfer effects while switching over.
- The Grabber card's AGC must first be configured to the new signal level.
- It is not possible to change channels from image to image. The user should adhere to inhibit times of at least 80 ms. This is dependant upon the signal sources connected.

## ② Selection of the luma notch filter for black/white -operation

**void Set\_BW(WORD nDevNo, WORD nOn);**

nOn: 0 = composite-signal at the input  
(activate luma notch filter),  
1 = b/w-signal at the input (deactivate luma notch filter)

If a black/white camera is connected to the Grabber, a luma notch filter is not necessary, which avoids disturbing color moiré from the brightness signal (*Cross-Color-Effect*).

This function allows the activation and deactivation of the luma notch filter by software. For b/w operation the deactivation of the luma notch filter is wise, because you can improve the sharpness of the image. In standard mode the luma notch filter is activated.

## ⇔ ② De-/Activation of the Interlaced Mode

**void Set\_Interlace(WORD nDevNo, WORD nInterlace);**

nInterlace:     0 = Non-Interlace  
                  1 = Interlace  
                  2 = Field Aligned

This function indicates to the Grabber, that the incoming videosignal is an interlaced or none interlaced signal. This will influence the vertical scaling filter. For a whole frame the option *Interlace* and for a field the option *Non-Interlace* should be selected in order to reduce artefacts from the motion.

When displaying only half frames, a 20 ms recess usually occurs between the two digitization events. This is because the other half frame cannot be displayed due to the half frame rastering delay. In the *Field Aligned* mode, the second half frame is internally moved over by a half row, so that the frame can fit onto the first field. This method allows a field to be returned in a 20 ms rhythm.

Because the half frame is altered during the electronic filtering, this function is only suitable in a range limited for measuring and automation tasks.



---

## ⇔ 2 De/Activation of the AGC

```
void Set_AGC(WORD nDevNo,WORD nCAGC, WORD nAGC,  
             WORD nCrush);
```

nCAGC: 0 = chroma AGC off  
      1 = chroma AGC on  
nAGC:  0 = AGC on  
      1 = AGC off  
nCrush: 0 = none-adaptive AGC  
       1 = adaptive AGC

The pciGrabber-4x4 contains two AGCs. The common AGC controls the signal level of the composite- or Y signal and controls correspondingly the input amplitude. In addition the chroma AGC controls the adaptation of the amplitude of the color signal. For the AGC the modulus *Adaptive AGC* is available. In this case the overflow bit of the A/D converter is monitored. If an overflow occurs, automatically the A/D reference voltage is increased, which causes an increase of the input voltage range.

In general, operating the pciGrabber-4x4 with a non-adaptive gain control will suffice.

In some applications the adaptive AGC might cause disturbances (e.g. this is the case when working with absolute brightness values). Then the non-adaptive AGC should be used.

When using applications that switch between multiple cameras (i.e. video monitoring), the switch time between cameras can be reduced, under certain circumstances, by using adaptive AGC.

### **Note:**

The standard AGC may **not** be switched off.

## ② De/Activation of the color killer

**void Set\_CKill(WORD nDevNo, WORD nCKill);**

nCKill: 0 = color killer off  
1 = color killer on

In case of b/w signal sources are connected to a color system, color noise can be created. The color killer eliminates this effect, by testing if the color burst is present and if necessary, deactivates the color evaluation.

It might be desirable to digitize a color signal with a weak color carrier, and the recognition of the color carrier is not practicable, so that only grey values are digitized. With *nCKill* = 0 the color killer is turned off, so that the image is digitized with color but it might have some color noise.

For the default, the color killer is on, and switching between color- and b/w-sources is done automatically.

## ❷ Dropping fields/frames in a video Signal

```
void TemporalDect(WORD nDevNo,
                  WORD nDecField,
                  WORD nFldAlign,
                  WORD nDecRat);
```

nDecField: 0 = drop frame(s)  
           1 = drop field(s)  
 nAlign: 0 = odd field will be dropped first  
          1 = even field will be dropped first  
 nDecRat: number of the fields / frames to be dropped out of 50 (PAL)  
           or 60 (NTSC)

The pciGrabber-4x4 allows for continuous digitization to select the number of images digitized per second. Default are 50 or 60 (NTSC) digitized images per second (video-standard).

With the help of this function it can be determined how many images of the 50 or 60 images are *dropped* during digitization.

The other two parameters give instructions of the kind of omissions: The parameter *nAlign* aligns the start of the decimation with an even or odd field.

The parameter *TDecField* defines whether decimation is by fields or frames.

Example (PAL/SECAM):

- *nDecField=0, nDecRate=2*

The reduction refers to frames. From 50 images two are omitted. The images 1-24 are produced as usually, then an image is omitted. Images 26-49 are produced and then again one image is omitted.

- *nDecField=1, nDecRate=25*

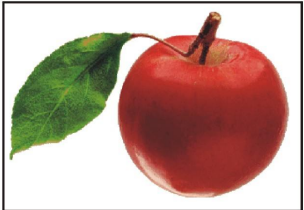
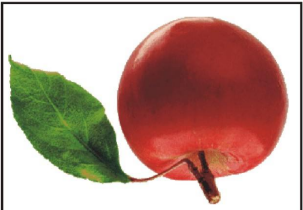
In this case 25 fields are omitted of 50. The result will be, that each second image will be produced, so always the same field type will be omitted. Which field will be omitted first, depends on *nAlign*.

## ☆ ❷ Flip Image Vertically

### DWORD FlipPicture(WORD nDevNo, unsigned char flip)

flip : 0 = image is stored upright  
 : 1 = image is flipped vertically (default)

With this function, the vertical orientation of the image in the memory can be set:

flip = 0	flip=1 (default)
<p>The image is stored upright. This means, the lowest image memory address contains the <u>upper</u>-left corner of the image:</p> <div data-bbox="231 922 726 1153">  </div>	<p>The image is flipped vertically before it is stored in the image memory. The lowest image memory address contains the <u>lower</u>-left corner of the image:</p> <div data-bbox="790 922 1284 1153">  </div> <p>This might be useful, if image data are processed in BMP-format.</p>

### Important:

- The setting of the vertical image orientation with *FlipPicture()* must be done before the *Set\_Image()* – function is called. The settings take effect not before *Set\_Image()* is called. To change the settings while the grabber is running, it has to be stopped first. Then the setting can be changed by calling the command sequence *FlipPicture()* and *SetImage()*.
- The default setting is **flip=1** (the image is stored upside-down; for compatibility reasons.)

## ② Setting the size and scaling of the image

```
void Set_Image (WORD nDevNo,
               WORD nOhpos, WORD nOvpos,
               WORD nOhsize, WORD nOvsize,
               WORD nOppl, WORD nOlines,
               WORD nOColformat,
               WORD nEhpos, WORD nEvpos,
               WORD nEhsize, WORD nEvsize,
               WORD nEppl, WORD nElines,
               WORD nEColformat,
               WORD nColSystem,
               WORD nInterlaced,
               WORD nSingleShot);
```

nOhpos, nOvpos : position of the left upper corner of the odd-section of the video picture  
(hpos = horizontal, vpos = vertical)

nOhsize : size of the odd-section in X-direction

nOvsize : size of the odd-section in Y-direction

nOppl : required size of the odd-video picture X-direction  
(ppl = pixel per line)

nOlines : required number of lines of the odd-video picture

nOColformat: required color format: (RGB32, RGB24, RGB16, RGB15, Y8, YCrCb 4:2:2, YCrCb 4:1:1)

nEhpos, nEvpos : position of the left upper corner of the even picture section of the video picture  
(hpos = horizontal, vpos = vertical)

nEhsize : size of the even-picture section in X-direction

nEvsize : size of the even-picture section in Y-direction

nEppl : required size of the even-video picture in X-direction  
(pixel per line)

nElines : required number of lines of the even-video picture

nEColformat: required color format: (RGB32, RGB24, RGB16, RGB15, Y8, YCrCb 4:2:2, YCrCb 4:1:1)

nColSystem : code for color system (see *Set\_Color\_System*)

nInterlaced : 0 = Non-Interlace

1 = Interlace

2 = Field Aligned

nSingleShot : 0 = continuous digitization

1 = **one** single image is grabbed

The routine ***Set\_Image*** () defines the size, the position and scaling of the image sections delivered by the Grabber separately for odd and even images. In addition, the data format in which the picture will be stored later in the memory, will be defined, and the address of this memory region is determined.

**Caution:**

Calling the function can only occur when the Grabber is in Stop mode. *Set\_Image* cannot be called when the Grabber is still in the digitization process, or when a Single-Shot digitization is not terminated by the *Stop\_Grabber* function.

The settings for both fields can be established separately and the parameters have a letter prefix 'E' = even image or an 'O' = odd image. Parameter without those letters are valid for both fields.

For both fields different sizes can be stated. They might be stored in different memory regions and can be processed in different ways.

In *Interlaced-Mode*, both fields are interlaced and are stored in a common memory region and have a maximum resolution of 720 x 576 pixels (PAL) or 640 x 480 pixels (NTSC), respectively. (See also the section *De-/Activating the Interlace Mode*)

In the following we consider the setting of the parameters without the field specifications (for example *hsize* for *nEhsize* / *nOhsize*). You have correspondingly to precede the letter 'E' for 'Even' or 'O' for 'Odd'. If a specific field is required, it will be indicated as such.

By specifying the parameters, a window with the size of the image is set (in pixels) first. This is achieved by the parameter *hsize* and *vsize*. *Hsize* indicates the number of pixels of the recorded image in x-direction, and *vsize* the number of pixels in y-direction.

The value *ppl* and *lines* define how many pixels should be generated from the incoming video picture. *ppl* indicates the number of pixels per line, and *lines* determines how many lines (pixels in y-direction) are produced. Both values indicate the resolution of the image.

A whole frame in PAL format has 720 pixel x 576 lines. In order to digitize the image with the highest resolution, *ppl* will be 720 and *lines* = 576. The smallest resolution for PAL is 50 x 40 pixels per frame.

If fields are used, the maximum resolution is 720 x 288 lines.

Since the definition of *ppl* and *lines* always refers to fields, the actual number of lines of the TV-picture to be digitized is twice as high. The width/height ratio for the digitized field (maximum 720 x 288 pixels) gives a distortion of two in y-direction. In order to avoid this effect digitization is done in interlaced mode (if the high resolution is required) or you can reduce the resolution to *ppl*=360, *lines*=288 and the resolution will be proportional with 360 x 288 pixels.

**Note:**

For stereometric work or automatization applications, a correct proportional resolution is not always required, as long as the distortion is taken into consideration in the algorithm. So the complete field resolution of 720 x 288 can be used for stereometric work, which is more accurate in x-direction than in Y-direction. It is also possible to adjust the axis of the camera in the direction the measurement has to be taken.

The window with the proportions *hsize* x *vsize* is a section of the digitized image, which has the size *ppl* x *lines*. If *hsize* = *ppl* and *vsize* = *lines* the whole digitized image is displayed. If this parameters are smaller, only a section of the image is displayed. The ratio of *hsize* to *vsize* does not alter the proportions of the image, since no scaling occurred and only a certain section of the image is taken. Figure 47 and Figure 48 demonstrate the significance of the parameter.

*vsiz*e and *lines* always must be represented in relation to a field in case a field was digitized.

If the section defined by *hsiz*e and *vsiz*e is smaller than the size of the area determined by *ppl* and *lines*, the window can be moved in the digitized image with the parameter *hpos* and *vpos*. For *hpos*=0 and *vpos*=0 the window is positioned in the upper left corner of the digitized image.

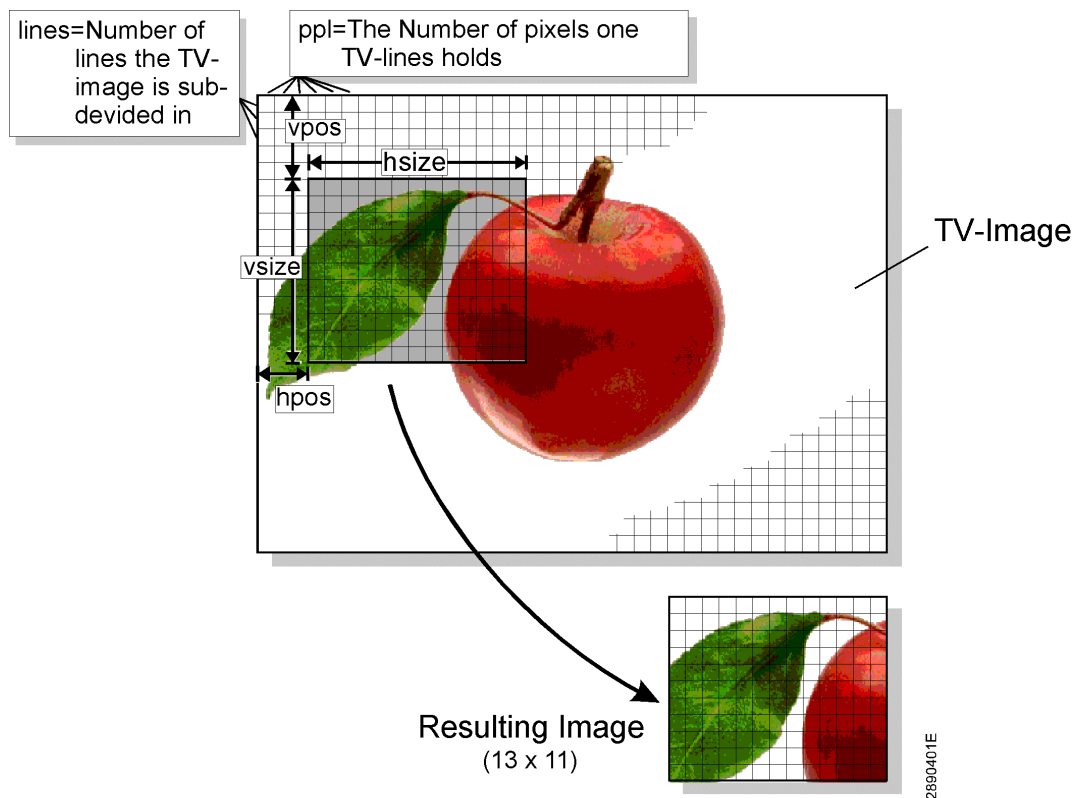


Figure 47: Scaling and Cropping



**Caution:**

- The area of the window defined by *hsize* and *vsize* and with the position *hpos* and *vpos* should never abandon the region of the digitized image defined by *ppl* and *lines* :

*hpos*=100, *hsize*=200, *ppl*=200 : not admissible since last 100 pixels are not defined

*hpos*=1, *hsize*=200, *ppl*=202 : admissible all pixels are in the image

*hpos*=100, *hsize*=100, *ppl*=200 : admissible

*hpos*=100, *hsize*=200, *ppl*=300 : admissible

*hpos*=300, *hsize*=300, *ppl*=800 : not admissible: image has more pixels than the TV-standard provides

(correspondingly in Y-direction)

- All parameters in horizontal direction must have even values.

(*ppl*=123 is not admissible, *ppl*=124 is admissible.)

- If all parameters, which influence the size of the image, are set to 0, then no field is produced.

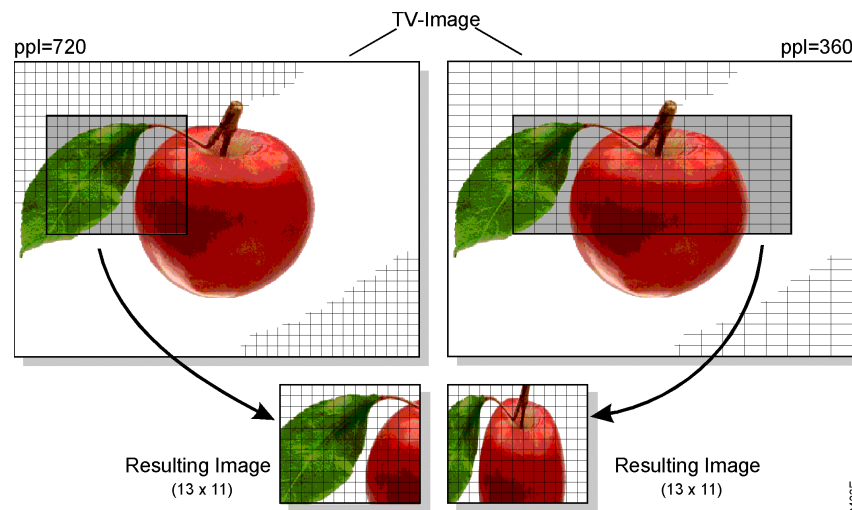


Figure 48: Example of Scaling: Only the *ppl* Value is Different

**Examples:**

- **field-based digitizing** A quadratic image of the size 256 x 256 pixels should be digitized with a resolution and proportion corresponding to the TV picture.

(a) Resolution and Scaling

For digitization a field (288 lines) is sufficient. In order to obtain a correct width/height ratio, the resolution in X-direction has to be reduced to the half (since field = half height).

Therefor  $720:2 = 360$ .

The result is :  $ppl=360$ ,  $lines=288$

(b) Size of the section

The image should have a quadratic size of 256 x 256 pixels.

The result is :  $hsize=256$ ,  $vsize=256$

(c) Positioning

It is advisable to center the section of the image.

In x-direction only 256 pixels of 360 pixels are displayed in the window. At the edge  $360 - 256 = 104$  pixels remain, which are divided in equal parts to both sides, which result in 52 pixels on both sides.  $hpos$  is the size of the left edge, so  $hpos=52$ .

Correspondingly in Y-direction:  $(288-256):2=16$ ;  $vpos=16$ .

**Note:**

It would be wrong to set  $ppl=256$  and  $lines = 256$ . In this case the width/height ratio (TV-Norm 4:3) would be changed to 1: 1 and the image would be distorted. It would be possible to set  $lines = 256$  and to compute  $ppl$  by the width/height ratio. In this case we would gain the optimal height of the image.

- ***field-based digitizing with zoom*** An image of the size 120 x 100 should be produced, for which the original image is magnified with the factor 2 in X- and Y-direction.

(a) Resolution / Scaling

It is sufficient to digitize a field. Without magnification 360 x 288 pixels are used. In order to allow the expansion we set 180 x 144 pixels :  $ppl=180$ ,  $lines=144$ . The width/height ratio is maintained.

(b) Size of the Section

Corresponding to the size of the window we set:  
 $hsize = 120$ ,  $vsize = 100$ .

(c) Positioning

With the parameter  $hpos$  and  $vpos$  the window section can be shifted  $180 - 120 = 60$  pixels in X-direction and  $144 - 100 = 44$  pixels in Y-direction.

- ***Full Frame Digitization*** A 700 x 500 image should be delivered with resolution and proportions that correspond to a TV image.

(a) Resolution and Scaling

For digitization, a frame is required. Since a full TV image serves as the foundation the image resolution results in  $ppl=720$  and  $lines=576$ . The lines value in the vertical direction must be evenly distributed to both half frames.

$$nOlines = nElines = \frac{1}{2} \text{ lines}$$

$$nOlines = \frac{1}{2} 576 = 288$$

$$nElines = \frac{1}{2} 576 = 288$$

Because the full image has a total of 576 rows, 720 pixels are required for the X-direction, so that the height to width ratio is maintained.

$$nOppl = nEppl = 720$$

#### (b) Cropped Image Size

The image should be 700 x 500 pixels large, resulting in an  $hsize = 700$ .

Again, to maintain the height to width ratio, the pixels must be evenly divided between the two half frames in the vertical direction.

$$nOvsize = nEvsize = \frac{1}{2} 500 \text{ Pixel} = 250$$

#### (c) Positioning

It is useful to center the image.

In the X-direction, only 700 of the 720 pixels are displayed in the window. Thus, a border of  $720 - 700 = 20$  pixels exists. The 20 pixels are evenly distributed on both sides, i.e. 10 pixels on the right and 10 pixels on the left.  $hpos$  is the size of the left-hand border, therefore  $hpos = 10$ .

$$nOhpos = nEhpos = 4 \text{ (even value)}$$

Correspondingly in the Y-direction:  $(288 - 250):2 = 19$ ;

$$nOvpos = nEvpos = 18 \text{ (even value)}$$

The parameter `nInterlaced` should be set to 1 so that the images are automatically interlaced.

After size and resolution of the image are defined, the data format has to be selected. The parameter *Colformat* describes the format, a pixel has to be stored in the memory of the PC, and how many Bytes are occupied by one pixel.

The format is determined by the application. In general three formats can be distinguished, which again can be separated in different formats. Figure 49 shows how pixels can be stored in the memory for different formats.

- **RGB :** The information for brightness is divided in three color channels: red, green and blue and are stored separately. This is a standard, which is used to process and handle color information.
  - For *RGB32* 32-bit, a double word per pixel is utilized. The lowest Byte of each double word contains the information for the blue color (8-bit), the second Byte the green and the third Byte the red color information. The highest Byte contains no information and is used only to obtain a whole double word for one pixel: After each double word a new pixel begins (see Figure 49 that related information have same hatching). The alignment of double words might have the advantage, that fast access commands could be used. The number of colors is 16 million ( $2^{38} = 16.777 \cdot 10^6$ ).
  - *RGB24* delivers the same information as *RGB32*, but does not contain the stuffing Byte. The image has the same resolution but occupies a smaller area of the memory.

- *RGB16* has a reduced resolution of the color. This format requires five bit for the blue and red channel, and the green channel has 6-bits. The color depth is 65.536 ( $2^5 \cdot 2^5 \cdot 2^6 = 65536$ ). One pixel needs 16-bits = 1 word. In Figure 49 the partitioning into three color channels of the word is depicted. The color information is arranged „left justified“; the lower bits are omitted, so that the color depth is reduced. The color depth of the green channel is twice of the two other channels. The RGB16-format corresponds to the color system of graphic cards with 65.535 colors.
- *RGB15*: The division corresponds to the RGB16-system, except that all color channels have the same color depth (each 5-bit=32 levels). Therefore we yield 32.818 colors. Altogether only 15-bits are necessary, so that the upper bit of a word is a stuffing bit and has the value 0 (see Figure 49).
- **YCrCb**: In this format we find grey values and color information separately. The parameter Y describes the brightness of the pixel (grey value) and the parameter (Cr,Cb) provides the color information. (Cr,Cb) can be considered as a vector of the chromatic circle. The tone of the color corresponds to the angle of the pointer, the saturation is presented by the absolute value of the vector. This format is applied for graphic cards with YCrCb-systems with separate processing of brightness and color value - which is very compact for the storage of images in memories.
- *YUY2*: This format corresponds to the format YCrCb 4:2:2.

- In one double word the information of two pixels is found. Y0 and Y1 is the information for the brightness of two adjacent pixels, Cb0 and Cr0 presents the color information of the first pixel, which is used for both pixels. The color information of the second pixel is not used.
- *BtYUV* : corresponds to YCrCb 4:1:1. Four pixels share one color information. The arrangement of the information in the memory is shown in Figure 49. Three double words are logically combined to one unit and contain the information for eight pixels. This is the value for the brightness for each pixel (Y0..Y7) and the color information of the first (Cb0/Cr0) and fifth pixel (Cb4/Cr4).

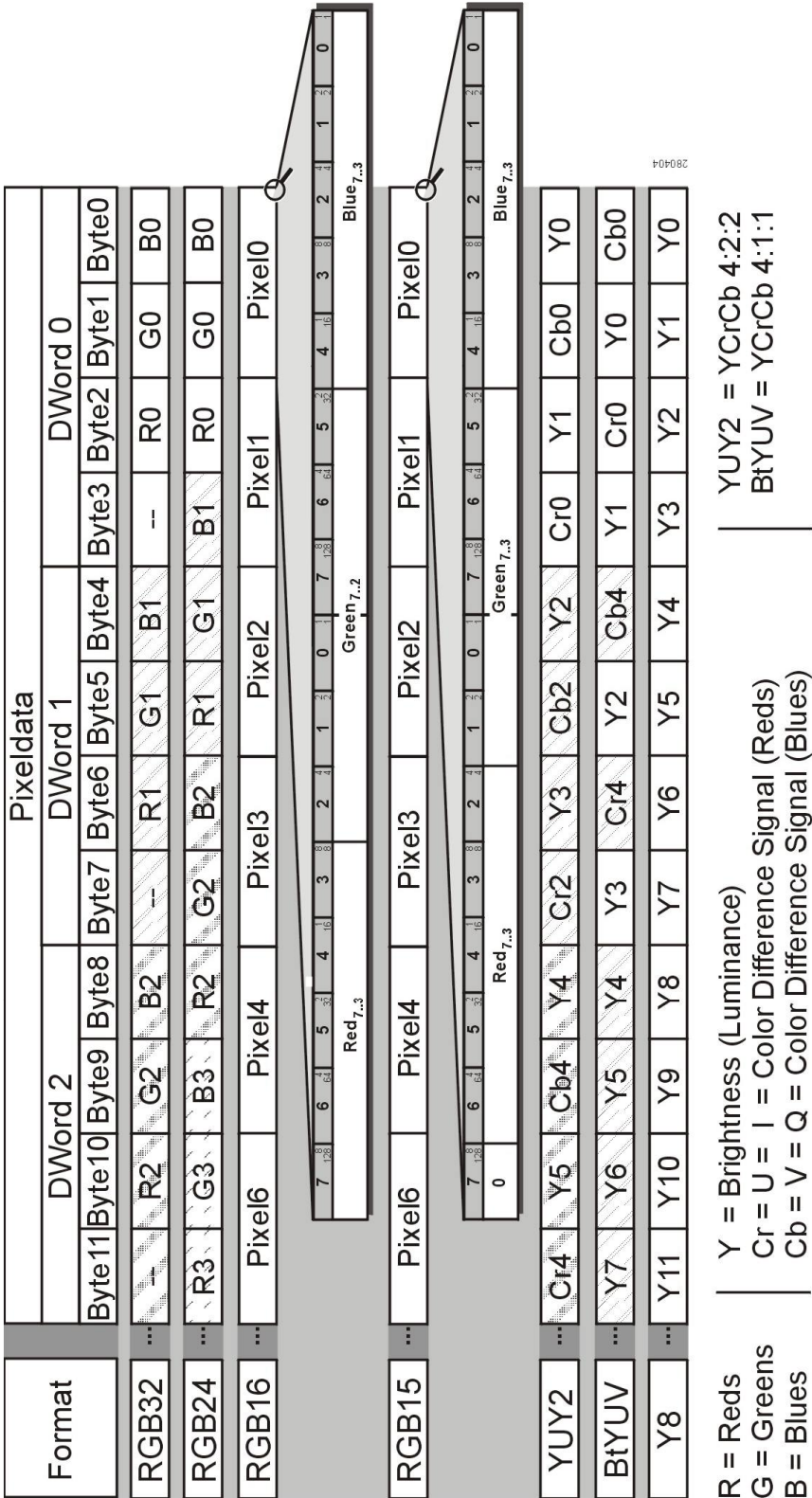


Figure 49: Color Format of the pciGrabber-4x4



- **Y :** In the grey scale format only the grey value, the value of the brightness of a pixel is stored. The color information is not considered. This format is recommended if color is meaningless for the evaluation.
  - **Y8:** In the Y8-format the grey value of each pixel is stored in sequence as a value with 8-bit, so that one Byte corresponds to one pixel.

Now the format of an image to be digitized, is exactly defined. The Grabber must now be instructed where and how to store the data of the image in the memory.

Therefore it is required to reserve a region in the main memory of sufficient size. This is achieved by utilization of the well known instructions for the allocation of memory (for example *malloc(...)*).

The Windows driver reserves an image memory space, in which the image is placed.

How much memory will be used? This will be calculated from the size of the image (number of pixels) and the required number of Bytes per pixel (color resolution):

Memory requirements per field =  $hsize \cdot vsize \cdot pixel\ size\ [Byte]$

The value *pixel size* can be depicted from *Table 7*.

For the format YUV2 and BtYUV it must be considered, that 2 or 8 pixels are combined logically and that the resolution of the image is selected correspondingly. The value calculated for the required region of memory is valid for **one** field (*even* or *odd*). These values must be added if a frame is required

Format	<i>pixel size</i> [Byte]
RGB32	4
RGB24	3
RGB16, RGB15	2
YUY2	4 Byte per 2 pixel
BtYUV	12 Byte for 8 pixel
Y8	1

Table 9: Required Memory Space of One Pixel for the Different Modi

If whole frames are required, because the resolution should be more than 288 lines, the Grabber can be instructed to store the frame in one single memory region. The Grabber automatically will interlace the two fields. If this option is required, you have to set *nInterlaced* = 1.

Finally the type of image recording, is described: With *nSingleShot* = 0 the Grabber is instructed to digitize continuously. That means, that after the start continuously digitized information are stored in the memory in real time (50 fields per sec.). In field mode (*nInterlaced* = 0) the information is stored to one memory region (20 ms), then the other region (another 20 ms) alternately. This means, that each field memory region is not accessed from the Grabber at least for 20 msec.

For frame mode (*nInterlaced* = 1) the Grabber stores continuously to the common memory, 20 msec the odd and then 20 msec the even lines.

During the evaluation of the image, it might be disturbing that the Grabber is just writing new data to the same region, and a mismatch might occur for fast moving objects. In this case a stop and go operation is recommended, or digitizing the frames in separate memory regions which are processed alternately.

*nSingleShot* = 1 has the effect that only one digitization takes place. Two fields are taken (one odd one even) or one whole frame. The user can record the images and with repeated starts new data are stored in the memory. This mode of operation is recommended, in case only occasionally images are digitized and no real time application is required.

In any case, if continuous or single shot grabbing is used: ***Set\_Image()*** configures, *how* the image is recorded. Grabbing is not started with this function but with the instruction *Start\_Grabber()* (see description below).

### ③ Start grabbing

**void Start\_Grabber(WORD nDevNo);**

The function *Start\_Grabber()* starts grabbing with the device specified by *nDevNo*. The result will be digitization with the beginning of the next available image. If continuous grabbing was selected, then one image after the other will be grabbed until the instruction *Stop\_Grabber()* is called. For single shot operation digitization is finished after one complete image.

*When does the first digitization take place?*

The driver handles whole frames. Always even/odd image combinations are evaluated. The timing will then depend on the desired field and the start up time currently applied field at the video input. We have to distinguish the following cases:

(1) An even-field is required to be digitized

a) At the input an even-field is applied:

Since the image just in process can not be evaluated anymore (the beginning is missing), the rest of the even field and the next odd field will pass and then digitization will start. So the next complete even field will be digitized.

The delay from the start instruction to digitizing will be < 40 msec.

b) At the input an odd field is applied:

The even field following the odd field will be digitized.  
Maximum delay time: 20 msec.

(2) An odd-field is required to be digitized

a) At the input an even field is applied

Since the driver evaluates even fields first, the rest of the incomplete even field and the following odd field will pass, until the Grabber will synchronize with the next even field.

Now the starting odd field will be digitized. The maximum delay will be < 60 msec.

b) At the input an odd field is applied

First the incomplete odd field will pass, and then the Grabber will be synchronized with the following even field, and will start the subsequent odd field. The maximum delay will be : < 40 msec.

### ③ Stop grabbing

**void Stop\_GrabberWORD nDevNo);**

This routine *Stop\_Grabber* aborts grabbing. The transfer of data will be cancelled immediately and the image might be incomplete.

#### **Caution:**

*Stop\_Grabber* must be called even for digitization in single shot mode (*nSingleShot* = 1 in *Set\_Image* ) when operation is finished automatically. The Grabber is locked (in standby mode), until *Stop\_Grabber* is called. Thereafter a new single shot can be taken with *Start\_Grabber* .

### ③ Show digitization status

**WORD Data\_Present(WORD nDevNo);**

return value: shows status of digitization  
(values 0 - 15 (4-bit) )

The function *Data\_Present* indicates if an even or odd image is stored in the memory.

In the separate bits of the returnvalue the status for continuous or single shot operation of the digitization is coded (see Figure 50).

Bit 0 and 2 indicate, that an even image was recorded, bit 1 and 3 represent an odd image. Bits 0 and 1 change their state with each advent of an even or odd image. For continuous grabbing this indicates when the content of the corresponding memory region is occupied completely with a new image. Each alteration of the status (0 to 1 and 1 to 0) indicates, that a new field was recorded.

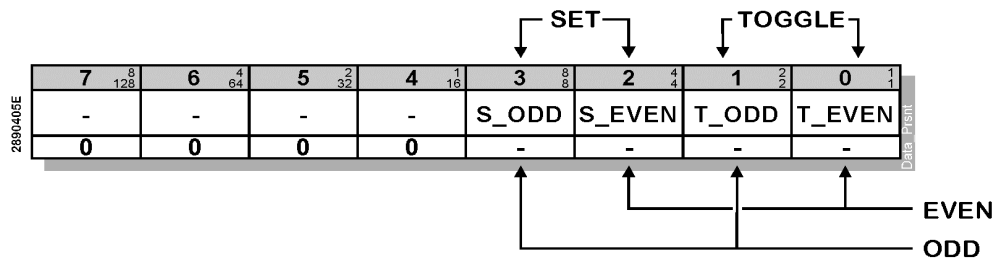


Figure 50: Return Values of ,Data\_Present‘

Bits 2 and 3 are set to 1, as soon as an even or odd image was completely recorded. Those bits are used, in case only one image should be taken (single-shot, defined by *Set\_Image* ). The bits remain set until a new digitization is started.

**Caution:**

Don't call the status too often during digitization, since each inquiry will occupy the PCI Express / PCI-bus, which might hinder the data transfer of the Grabber. You might include delay times between inquiries, in order to avoid slowing down digitization.

Please pay attention to evaluate the correct bits, which correspond to the actual mode, since otherwise your program might access the data during the wrong time interval.

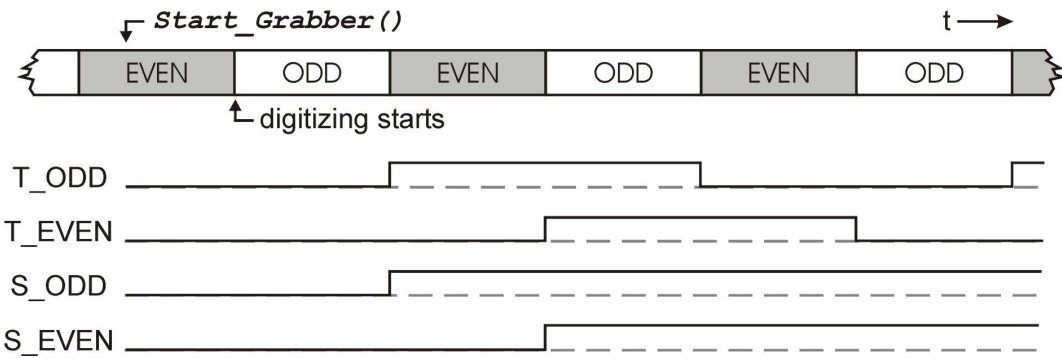


Figure 51: Timing Diagram of the Return Parameter of ,Data\_Present()

### ③ Reading image data

**DWORD GetPictureBufferAddress (WORD nDevNo, DWORD dwBitsSize) ;**

dwBitsSize: size of the memory region used for the image

return value: address of the beginning of the memory region

In order to read the digitized data from the memory region controlled by the VxD-driver, the user program must know the start address of the memory region.

The function *GetPictureBufferAddress* returns this address to the user program, at which the region of the memory starts, defined before by the VxD. In this memory the Grabber stores the data of the digitized image with the following structure:

(a) Only even- or only odd-fields are digitized

(the parameters of the dimension of the other field are zero)

⇒ The digitized field is placed at the beginning of the memory region of the image.

(b) Even- and odd-fields are digitized

(nInterlaced=0)

⇒ The even-field is placed at the beginning of the memory region. Adjacent to the even fields, the odd field is positioned. The start address of the odd- field is also the start address of the memory region of the image plus *nEhsize*; *nEvsize* the number of Byte per pixel.

(c) Frames are digitized in interlace-modus

(nInterlaced=1)

⇒ The digitized frame is composed of interlaced fields and begins at the start address of the memory region of the image.

The parameter *dwBitsSize* defines the whole size of the required memory region of the image. This is calculated from the number of pixels and the number of pixels per Byte: *hsize* · *vsize* · pixel size.

Please pay attention that *dwBitsSize* is the size of the **complete** memory region required for the image. If even- and odd-fields are digitized separately, you have to add the memory region for both fields. More information on calculating memory requirements can be found under the command *SetImage()*.

### ③ ☆ **Activate Interrupt**

**DWORD ActivateFieldInterrupt (WORD nDevNo,  
HANDLE \*hEvent);**

HANDLE: Pointer to an Event

Monitoring the progress of image digitizing can also be done using the hardware-interrupt of the framegrabber. An interrupt can be generated at the end of every field.

The interrupt is signaled to the software by the generation of an event. Waiting for this event is a method to monitor the digitization progress without polling the status flags (see *DataPresent()* ).

The function *ActivateFieldInterrupt()* sets up the interrupt configuration of the grabber card and activates the hardware interrupt. It creates the event and returns a pointer (handle) to this event.

You can monitor this event by using the Windows API-function *WaitForSingleObject()*. Please note, that the event has to be reset by calling the function *ResetEvent()*. Otherwise, new events can not be signaled.

#### **Important:**

- The event is signaled every time the digitizing of a field is finished. This happens only, if the digitization has been started (*Start\_Grabber()* ). After the occurrence of the field end, the signal remains active until it is reset by calling *ResetEvent()*. This enables



the application software to determine event some time later, whether the event had happened or not

To reset the event, the function *ResetEvent()* must be called.

Otherwise, the next call of *WaitForSingleObject()* would return immediately, because the signal is still active.

*ActivateFieldInterrupt()* should therefore be called only when the grabber is not running. It might be advisable to call *ResetEvent()* after that, to reset any older active signals, which might have been created in the meantime.

- To determine, which field had been digitized, the *DataPresent()* – function should be called immediately after the event had been signaled. In case of continuous digitization this is done by evaluating the state of the T\_EVEN / T\_ODD – flags. Care must be taken to read out this flags immediately, because their state will only persist for 20 after the end of the field. (This is, because 20 ms later, the next field digitization is completed and therefore the flags are updated).

*Hint:* An exclusive-or – combination of T\_EVEN and T\_ODD will result the parity of the field (1=ODD, 0=EVEN):

*field\_ready* := T\_EVEN ^ T\_ODD ;

- The event is created by the grabber driver with a very short time delay. Please notice, that the delivery to the application by the operation system might last a noticeable (and variable) time.

### ③ Checking video signal quality

**WORD Get\_Signal\_Status(WORD nDevNo);**

return value: 0 = no video signal at input  
1 = undefined  
2 = video signal present  
3 = video signal present and line locked

With the help of this function it is recognized, if a camera is connected to the selected channel. In addition the quality of the signal can be evaluated.

The returnvalue 0 indicates, that no source is connected to the selected channel. If no synchronizing pulses are detected for 31 lines, it is anticipated, that no source is applied.

If the returnvalue is two, a source is applied. Interference at the input can cause a wrong indication.

The returnvalue 3 indicates a stable videosignal, which stems with high probability from an image source. This value is generated in case the horizontal synchronizing pulse is found within  $\pm 1$  clock of the expected position. This must be given for 32 lines in sequence. Vice versa the „HLOCK“ status will be not indicated, in case this criteria is not fulfilled for 32 consecute lines.

Therefor this indication will be very reliable. Some image sources such as a videorecorder tend to not having a stable timing. For such devices it might be possible, that the returnvalue 3 will never be indicated. Nevertheless digitization will be faultless by using the ULTRALOCK™-synchronization.

### ③ ☆ **Number of Processed Digitized Images**

With the pciGrabber-4x4, it is possible to count the number of digitized images. The following two functions are used for this purpose:

These functions are not compatible with the pciGrabber-4.

#### **BYTE Get\_CaptureCounter (WORD nDevNo)**

return value: Number of grabbed fields module 256

The function returns the number of digitized fields. The result is a Byte value and the counter jumps from 255 to 0.

#### **void Reset\_CaptureCounter (WORD nDevNo)**

Calling this routine sets the field counter back to zero.

#### ④ Setting the brightness of the image

**void Set\_Brightness(WORD nDevNo, short nBright);**

nBright:            brightness of the image (-128..127)

This function specifies the value in the register for the brightness in the video processor. The value is a constant added to the brightness of a single pixel. This brightness can be varied from -50 % to +49.6 %. One LSB corresponds to a change in brightness of 0.39 %:

$$nBright = \text{brightness}[\%] \cdot 2.5601 [1/\%]$$

#### ④ Reading the brightness setting

**short Get\_Brightness(WORD nDevNo);**

return value: Content of the register holding the value for brightness in the video processor.

With this function you can ascertain the actual brightness parameter of the video processor.

#### ④ Setting the contrast

**void Set\_Contrast(WORD nDevNo, WORD nContr);**

nContr:            contrast (0..511)

This function specifies the contrast of the image. The contrast is a constant factor, which is multiplied in the video processor (corresponding to the scaling) with the brightness of the pixel. The factor has a range of 0 % to 236.57 % :

$$nContr = \text{contrast} [\%] \cdot 2,1598 [1/\%]$$

#### ④ Reading the contrast setting

**WORD Get\_Contrast(WORD nDevNo);**

return value: actual contrast value

With this function you can ascertain the actual value of the contrast register of the video processor. The returned value is an integer number.

#### ④ Setting of the color saturation

**void Set\_Saturation(WORD nDevNo,WORD nSat\_U,  
WORD nSat\_V);**

nSat\_U: Saturation of the U-color portion (0..511, Default = 254)

nSat\_V: Saturation of the V-color portion (0..511, Default = 180)

This function allows the separate setting of the color saturation for the U- and V-color portion. With this parameters the amplification can be separately regulated for both color portions. Usually the relation of the values *nSat\_U* and *nSat\_V* are equal. The control of the difference between the U- and V- portion allows the elimination of color cast (which might stem from unbalanced color cameras). The resulting effect will be a change of the color of the image.

$nSat\_U = \text{U-saturation [\%]} \cdot 2.5400 [1/\%] ; 0\% \dots 201.18 \%$

$nSat\_V = \text{V-saturation [\%]} \cdot 2.1396 [1/\%] ; 0\% \dots 238.83 \%$

#### ④ Reading the color saturation

**WORD Get\_Sat\_U(WORD nDevNo);**

**WORD Get\_Sat\_V(WORD nDevNo);**

return value: value of the actual U- or V-color saturation

This function provides the content of the registers for the color saturation.

#### ④ Correction of the hue (NTSC only)

**void Set\_Hue(WORD nDevNo, short nHue);**

nHue: hue, phase position of the color signal (-128..127)

With this function you can control the hue for the digitization of NTSC- color images, which is accomplished by changing the phase position of the color signal. For PAL this value is insignificant since phase errors are automatically compensated.

One LSB corresponds to a correction of the phase angle of  $0.7^\circ$ , therefor the color signal can be varied in the range of  $-89.3^\circ$  to  $+90^\circ$

This value must be set to 0 to ensure proper functioning of the color decoder under PAL.

#### ⇔ ④ Reading the content of the register for hue

**short Get\_Hue(WORD nDevNo);**

return value: value of the phase position of the color signal

This function provides the hue value.

#### ④ De/activation of the luma-low-pass-filter

**void Set\_LDec(WORD nDevNo, WORD nOn, WORD nHFilt);**

nOn: 1 = Luma decimation on  
0 = Luma decimation off

nHFilt: 0 = automatic filter selection  
1 = CIF filter  
2 = QCIF filter  
3 = ICON filter

For small image sizes, a higher quality is achieved, if the resolution of the input signal is reduced (so that sharpness of the image is adjusted to the resolution of the displayed image). For this reason this function is able to insert an optional low-pass-filter into the luma path. With the parameter *nHFilt* the used filter is adapted to the size of the image:

The function *automatic filter selection* adapts the filter setting to the size of the image. (see also *Set\_image*). In addition the filter can be adjusted manually to one of the standard formats CIF (= 1/2 whole frame), QCIF (1/4 whole frame) and ICON (1/8 whole frame). Default: Luma-low-pass is turned off.

#### ④ De-/Activation of the test image

**void Set\_ColorBars(WORD nDevNo, WORD nColorBars);**

nColorBars: 0 = test image off  
1 = test image on

This function controls the activation of a test image. The test image contains vertical colored bars. The test image is independent of an input signal. In order to see the whole image the size of the image should have CIF-format.

#### ④ Adjustment of the range of values

**void LumaControl(WORD nDevNo,  
WORD nRange,  
WORD nCore);**

nRange:    0 = luma range 16 - 253  
             1 = luma range 0 - 255  
nCore:     0 = 0 all brightness values are transmitted  
             1 = 8 all brightness values <= 8 are interpreted as 0  
             2 = 16 all brightness values <= 16 are interpreted as 0  
             3 = 32 all brightness values <= 32 are interpreted as 0

With this function the output format of the brightness and color value can be adapted to the application

The parameter *nRange* determines the range of values for the brightness (permissible grey values).

- *nRange*=0 corresponds to the standard range of values specified in CCIR 601. The range of values for the brightness is restricted to the values 16 to 253 where Y=16 corresponds to black. The range of values for colors is 2...253 with Cr/Cb=128 as zero (signed).
- *nRange*=1 allows the utilization of the whole range, that is for Y the range 0...255 with 0=black, the chroma range is defined as for *nRange*=0.



## ⑤ Reading/Writing data via the option port (GPIO-Port) Reading the jumper settings

8 I/O pins are available on the pin header row option port. These pins can be controlled with the following three functions:

**void Set\_GPIO\_Direction (WORD nDevNo, WORD nDirection);**

**void Set\_GPIO\_Data (WORD nDevNo, WORD nData);**

**WORD Get\_GPIO\_Data (WORD nDevNo);**

nDirection: can have values between 0 and 4096  
(direction control of the board)

nData: Data, which should be output via the option port

return value: Data, which are read from the option port

The bit from 1 to 8 are the I/Os of the Option Ports and Bit from 9 to 12 the jumpers.

The pciGrabber-4x4 has an option port with pins, which can be used separately to read or write digital signals. With the following functions the option port is controlled. You can define which pins will work as input or output, set which output pins have low or high level and determine what level is applied to the input pins.

With the help of the function **Set\_GPIO\_Direction** each pin of the port can be configured as input or output. For this purpose the lower 8-bits of the parameter *nDirection* are evaluated. If a bit is set to 1, the corresponding port is configured as output. A 0 results in a configuration as input.

### Caution:

Please pay attention, that a pin of the port is only switched to output, when no external signal is applied to the pin. Otherwise it might happen, that the pin circuitry is damaged.

All pins are configured to input when starting the computer. Please ensure that the pins are high ohm and therefor the logic level is not

defined. Hardware precautions (i.e. pull-up resistors) must be taken in order to determine the behavior of controlled components during the start-up of the computer until the GPIO port has been configured.

With **Set\_GPIO\_Data** each port pin is an output with the level setting „High“ = 1 or „Low“ = 0. nData is a 12-bit value, whereas to each bit one pin is assigned. The setting is only effective for those pins, which are configured as output.

The function **Get\_GPIO\_Data** reads the data from the port pins, which had been selected as input and returns a 12-bit value. The return value for the pins configured as output, will be the actual setting.

## ⑤ ☆ Transmitting Data via the I<sup>2</sup>C Interface

The user can implement these functions in order to read and write to devices connected to the I<sup>2</sup>C interface.

### Caution:

The I<sup>2</sup>C-EEPROM, found on the Grabber card, is protected against accidental writing. Therefore, access to the device address space 0xA0 to 0xA3 is not allowed.

In order to obtain access to the internal EEPROM memory space, please use the appropriate special functions.

**void I2C\_Set\_BR\_Mode (WORD nDevNo, BYTE bMode)**

bMode	baudrate
	pciGrabber-4x4: 0 = 99,2 kHz, 1 = 396,8 kHz

This function determines the baud rate for transmission on the I<sup>2</sup>C bus. A lower or a higher transmission rate can be selected.

**BYTE I2C\_ReadByte (WORD nDevNo, BYTE bChipAddress,  
BYTE bSubAddress, BYTE \*bByteRead)**

bChipAddress: Device address for the I<sup>2</sup>C device on the bus  
bSubAddress: Internal memory address for the I<sup>2</sup>C device  
\*bByteRead: Pointer points to a Byte variable. Result  
is written into the Byte variable.

return value: SUCCESS, NOACK, INVALID\_ADDRESS

*I2C\_ReadByte* is used to read a Byte from the memory space of an I<sup>2</sup>C device. The result is given in a variable Byte type. This Byte must be defined before hand.

The function returns an error code as a return value. NOACK indicates that no I<sup>2</sup>C device has been registered under the given device address. INVALID\_ADDRESS indicates an unsuccessful attempt to access the protected area of the EEPROM mounted on the Grabber.

**BYTE I2C\_WriteByte (WORD nDevNo, BYTE b ChipAddress,  
BYTE bSubAddress, BYTE bData)**

bChipAddress: Device address of the I<sup>2</sup>C device on the bus  
bSubAddress: Internal memory address of the I<sup>2</sup>C device  
bData: Byte written into the specified address

return value: SUCCESS, NOACK, INVALID\_ADDRESS,  
WRITE\_FAILED

Writes a Byte into the memory space of a specified I<sup>2</sup>C device. The function returns an error code as a return value. After writing, the function reads the string and checks to make sure that the written and read values are identical. If the values are not identical, then the error code WRITE\_FAILED is returned.

**Note:**

For registers that have different functions in read and write accesses (i.e. status / command), most of the time the read value does not match the written value. In this case, WRITE\_FAILED is returned, although the write operation was successful.

## 5 ☆ Using Internal EEPROM

The pciGrabber-4x4 has per decoder an internal non-volatile memory. This memory is intended for the storage of parameters. A total of 252 Bytes is available to the user.

### Note:

Since the predecessor models to the pciGrabber-4x4 do not include internal memory, the following functions are not compatible.

### **BYTE I2C\_ReadEEProm (WORD nDevNo, BYTE bSubAddress, BYTE \*bByteRead)**

bSubAddress: Memory address to be read (0x00 ... 0xFB)

\*bByteRead: Pointer points to the Byte variable. The result is written into the Byte variable.

return value: error code = SUCCESS, NOACK

Reads a Byte value from the EEPROM. Specified by calling the memory address that is to be read. The result is returned in a Byte variable form that must be pre-defined.

The function returns an error code as a return value (*see I2C\_ReadByte*).

### **BYTE I2C\_WriteEEProm (WORD nDevNo, BYTE bSubAddress, BYTE bData)**

bSubAddress: memory address written to (0x00...0xFB)

bData: Data byte that is written

return value: Error code = SUCCESS, NOACK, WRITE\_FAILED

Writes a Byte into the internal EEPROM. Specifies the desired memory address and the stored Byte to be written to. The function returns an error code (*see I2C\_WriteByte*).

### Note:

The life span of the internal EEPROM memory is 1 million write accesses. The number of read accesses is unlimited

## ⑤ Direct access to the video processor's registers

**WORD Read\_Local\_DWord(WORD nDevNo,  
WORD nRegister\_Number,  
DWORD \*lContent);**

nRegister\_Number: number of the register  
lContent: contents of the registers

**WORD Write\_Local\_DWord(WORD nDevNo,  
WORD nRegister\_Number,  
DWORD lContent);**

nRegister\_Number: Number of the register  
lContent: data of the register

Almost all the functions of the pciGrabber-4x4 can be controlled about the routines of the driver. We expressly recommend the use of the standard functions.

For the case that the user would like to affect directly the registers of the pciGrabber these two functions are available. With these both functions it is possible to read or write all registers of the decoder. Should you need further information about the registers of the pciGrabber, please turn to the PHYTEC support.

### **Attention!**

Some registers of the pciGrabber-4x4 must be set in the present device configuration with certain values to guarantee the function of the card. Check hence the exact meaning of the register before you change them.

PHYTEC cannot assume liability for damages which possibly originate from manipulation of the registers.

## **7 Trouble-Shooting**

- The color representation is very much reduced in the Windows-demo-program.
  - Check the configuration of the graphic card. In order to yield the full resolution of color of the pciGrabber-4x4, the graphic card must be set at least to 16 Mio. colors.
- Only a blue image is displayed.
  - The selected input is not connected to a video source. Usually a blue image is shown. Please check if the correct input channel is selected.
- The digitized image shows streaks and stripes
  - It might be a Moiré-effect caused by the color signal. Check if the luma notch filter is enabled.
  - The cable to the camera might be defect (check shielding).
  - A ground loop might be existent by an additional ground connection between PC and camera causing a mains hum. Check, if ground loops or power supply cause a mains hum .
- At the S-video input no image is digitized.
  - Did you connect the S-video input properly?
  - Did you chose the right decoder in the menu Basic Settings?
  - Are both S-Video inputs connected (only one's possible)?
  - Was the pciGrabber-4x4 configured to the S-video operation?
  - Has an incorrect input socket been selected (Mini DIN / Combi)?

- Has another channel been selected after switching over to S-Video with Set\_Channel?
- For S-video operation the image is only black/white
  - Is the chroma-signal properly applied?
  - Is the Grabber configured to S-video operation?
- For S-video operation the image is black/white with color disturbances.
  - Is the chroma-signal properly connected?
  - Is a S-video-source connected (not a composite)?
- The image is displayed incompletely/ or very slowly
  - Increase the delaytime between status inquiries. The PCI-bus might be blocked by too many calls concerning the status.
  - Was the resolution reduction concerning to time used?
- The image has no contrast / too bright / too dark
  - Check the setting of brightness, contrast etc.
  - If necessary activate the AGC
- The image skips or the fields are mismatched
  - The time delay during switching the channels was too short.
- The image appears without color / with the wrong format
  - Is the appropriate color system selected?
  - Was the Grabber correctly initialized?
  - Was the delaytime after channel switching long enough?



- The lower edge of the image is missing/no image is digitized
  - The vertical image size was chosen too large.
  - It was not recognized, that starting from a line number higher than 288 lines 288 (PAL) or 262 (NTSC), a whole frame must be used.
- The colors are not presented properly.
  - The values *hpos* and *hsize* must be even, in order that the color decoding works properly.
  - In the user program: You mixed up Cr/Cb .
  - Did you select the correct color format?
  - The chroma gain registers were changed.  
Please pay attention, that for a correct Hue setting the U- and V-component must be identical in respect to their *percentage* value. But the values of the *registers* are different!
- After a standby of the system the pciGrabber-4x4 lost the functional capability.
  - The pciGrabber-4x4 does not support the standby except for the use of a mainboard which does not power down the 3.3 Volt supply on the PCI Express-bus during a standby. You must reboot the system if the 3.3 Volt power down during the standby.
- During continuous grabbing the image jumps one line up/down.
  - The parity of the field is neglected. Pay attention in order to demand always the same field. The memory region for the even and odd image should not be the same!

**Note:**

The mismatch is actually a half line, therefore the mismatch can not be compensated by displacing one line.

- The video source delivers no proper signal.
- Switching channels between two cameras occurs too quickly

- During the display of frames diagonal lines/circles are interlaced or have unclear perimeter edges.
  - Even and odd fields are interchanged (only possible in case you use single fields).
- After digitization an image, no further digitization is possible.
  - Before requesting new digitization, the last operation must be finished with the call of the function *Stop\_Grabber()* .
- In frame mode, using continuous monitoring, a shadow effect is recognized, despite the Grabber digitizing two complementary fields in real time.
  - The effect is due to the slow speed of the host PC displaying the image. The update at the screen is not fast enough, even through the images are found digitized in the main memory.
- The supply voltage output for the camera is not functioning.
  - Check the mini fuse .
  - Is a power supply cable for the PC network device connected to the plug connector on the Grabber at X7? (small floppy power supply plug 5 V/GND/GND/12V)
  - Check the proper connection of the cable: The supply voltage is only available at the lower HD-DB15-socket. If the cable is connected to the upper plug, the power plug is connected to the video input 5.
- The displayed image is disturbed by horizontal stripes, which might show parts of the preceding image. For moving objects horizontal lag effect occur.
  - The Grabber is not able to transfer the image data in real time via the PCI-bus, since other cards on the bus stress the bus too much, or the bus-configuration of the BIOS is not correct.

Please check the configuration of the other PCI-cards and the configuration of the BIOS.

## Index

„	
„LabView“ driver .....	24
<b>A</b>	
accessories .....	5
<b>Arithmetic Operations for</b>	
<b>Static Images</b> .....	59
<i>Arithmetics</i> .....	59
AUTO function	
Driver .....	101
<b>B</b>	
Basic parameters .....	38
BNC plug .....	30
both fields .....	51
<b>C</b>	
color bars .....	59
<i>Color Bars</i> .....	59
<i>Color Meter</i> .....	58
<i>Color Mode</i> .....	42
comb effect .....	51
<b>Compatibility to the</b>	
<b>pciGrabber-4</b> .....	91
composite inputs .....	28
Composite Inputs .....	15
<b>Composite Sources</b> .....	49
connecting composite sources ..	30
<b>Connectors</b> .....	9
Cross Hairs	
Blending In/Out .....	56
<b>D</b>	
<b>Data Format</b> .....	7
delivery .....	4
Demo program	
Image Settings .....	48
Demo Program	
Channel Selection .....	48
Image Resolution .....	49
Image Selection .....	49
Installation .....	25
Operations .....	37
Demo Programing	
Normalizing .....	60
developing environments .....	79
Device driver	
WIN 98/ME .....	22
Win NT 4.0 .....	23
DLL .....	85
Driver	
Configuring Composite Inputs	
.....	99
Configuring Input Channels	102
Configuring the S-Video Mode	
.....	100
De-/Activating the Interlaced	
Mode .....	104
Defining the Version Number	
for the DLL (Win) .....	93
Grabber Name Read as a Clear	
Text String .....	97
Using Internal EEPROM .....	141
Using the I2C Interface .....	139
Windows 98 / NT .....	84
driver installation .....	22
<b>E</b>	
EEPROM .....	141
<b>F</b>	
<i>Field Aligned</i> .....	104
<i>Frame Rate</i> .....	44
Full Frame Digitization .....	115
Functions	

Classification .....	89	TemporalDect() .....	107
Data_Present() .....	125	Funktionen	
Get_Brightness .....	132	Get_Video_Status().....	99
Get_CaptureCounter() .....	131	Write_Local_DWord().....	142
Get_Contrast() .....	133	<b>G</b>	
Get_Error().....	92	<i>Gr4CDLL.DLL</i> .....	86
Get_GPIO_Data() .....	137	Grabber Card	
Get_Hue() .....	134	Installation.....	20
Get_Sat_U() .....	133	<b>H</b>	
Get_Signal_Status().....	130	half frames.....	50
GetVersionNumber (WIN) ....	93	half image.....	104
I2C_ReadByte.....	140	histogram.....	56
I2C_ReadEEProm().....	141	<b>I</b>	
I2C_Set_BR_Mode().....	139	I/O-Port	
I2C_WriteByte() .....	140	testen (Demoprogramm) ..	62, 63
I2C_WriteEEProm() .....	141	I <sup>2</sup> C interface .....	8
LumaControl() .....	136	I2C Interface	
Max_Device_Number() .....	94	Programming.....	139
Read_GrabberInfo().....	96	I <sup>2</sup> C interface .....	17
Read_Local_DWord() .....	142	<b>Image Resolution</b> .....	7
Read_OrderCode().....	97	IMAQ .....	24
Reset_CaputerCounter .....	131	<b>Information on Calling-Up</b>	
Set_AGC() .....	105	the Installed Grabber.....	96
Set_Brightness() .....	132	<b>L</b>	
Set_BW() .....	103	live image .....	37
Set_ChannelEx() .....	102	<b>Live Image</b> .....	55
Set_CKill().....	106	<b>M</b>	
Set_Color_System().....	98	measuring color values .....	58
Set_ColorBars() .....	135	<b>N</b>	
Set_Composite() .....	99	Number of Processed Digitized	
Set_Contrast() .....	132	Images .....	131
Set_GPIO_Data().....	137	<b>O</b>	
Set_GPIO_Direction.....	137	<i>Open Image on Start</i> .....	55
Set_Hue() .....	134		
Set_Image() (Win) .....	109		
Set_Interlace().....	104		
Set_LDec().....	135		
Set_S_VideoEx() .....	100		
Set_Saturation() .....	133		
Start_Grabber() .....	123		
Stop_Grabber() .....	125		

**P**

PCI slot ..... 20

Pinout ..... 13

**Ports** ..... 8

**Power Supply** ..... 6

**R**

Reducing Noise Levels ..... 62

Replacement fuse ..... 5

**S**

similar television technology .... 50

**Single Image** ..... 55

**Snapshot** ..... 55

Snapshots ..... 55

Standby ..... 145

Storing Images

    Demo Program ..... 64

Storing Parameters ..... 141

**S-Video Source** ..... 49

**Synchronization** ..... 7

**T**

**Twain driver** ..... 24

*Type Casting Settings* ..... 61

**V**

video source connections ..... 28

video sources ..... 26



**Document:** pciGrabber-4x4  
**Document number:** L-720e\_0, January 2009  
**How would you improve this manual?**

---

---

---

**Did you find any mistakes in this manual?** page

---

---

---

---

**Submitted by:**

Customer number: \_\_\_\_\_

Name: \_\_\_\_\_

Company: \_\_\_\_\_

Address: \_\_\_\_\_

\_\_\_\_\_

**Return to:**

PHYTEC Technologie Holding AG  
Postfach 100403  
D-55135 Mainz, Germany  
Fax : +49 (6131) 9221-33

Published by

**PHYTEC**