eGrabber-4*plus*

Hardware Manual

Edition June 2004

	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 order@phytec.de	1 (800) 278-9913 sales@phytec.com
Technical Support:	+49 (6131) 9221-31 support@phytec.de	1 (800) 278-9913 support@phytec.com
Fax:	+49 (6131) 9221-33	1 (206) 780-9135
Web Site:	http://www.phytec.de	http://www.phytec.com

4th Edition June 2004

Contents

Index of Figures and Tables

Contents

Part 1

Installation and Start-Up

1 Delivery Contents/ Technical Data

The eGrabber-4*plus* is a frame grabber card designed for use with industry standard PC/104-plus CPU-cards.

The card is available in various configurations:

Order-Nr.	Composite-Inputs	S-Video-Input	Integrated Power Supply
EPC-032	3	1	No
EPC-032-X1	9	1	No
EPC-032-X2	3	1	Yes
EPC-032-X1-X2	9	1	Yes

Table 1:   eGrabber-4plus Model Variants

Refer to the documentation included with the eGrabber-4*plus* to see which model you have. This will enable you to achieve optimal settings for your Framegrabber card.

The eGrabber-4*plus* includes the following upon delivery:

- eGrabber-4*plus* board in PC/104-plus format
- Installation CD with
  - ❍ Demo software
    (Windows' 95/98/ME/XP, NT4.0 and Windows' 2000)
  - ❍ Driver library for DOS (with DOS4GW)
  - ❍ Driver software for Windows' 95/98/ME/XP, NT4.0 and Windows' 2000
  - ❍ Twain driver for applications with Twain interfaces
  - ❍ Labview driver for photo processing applications using Labview (National Instruments, IMAQ – packet is required)
- the eGrabber-4plus manual

1.1 Accessories

The following eGrabber-4*plus* accessories may be ordered from PHYTEC:

- SMB to BNC connector cable for connection of a color camera with BNC-plug: length aprox. 2 m – Order number WK023

- S-Video- connector cable for connection of a color camera with a 4-pin Mini-DIN-plug (S-Video-output) length aprox. 2 m - Order number WK051

- Replacement fuse 2A T TR5 power supply (receptacle F1) – Order number KF012

Figure 1:   Accessory Cables

1.2 Technical Data

**Physical
Dimensions:** PC/104-plus Format

Data Bus: PCI bus 5 V, Master slot required
(PCI Rev. 2.1 compliant)
(PC/104-plus Version 1.0 compliant)

Power Supply: +5 V (250 mA idle, 300 mA digitizing)
(taken from the PCI bus)

Inputs: 3 or 9 (depending on the model, *see section 1.5.1*)
Composite video inputs, 75 Ω, 1 V_{ss}
1 S-Video input 75 Ω (0,7 V_{ss} / 0,3 V_{ss})

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

Synchronization: Composite sync. or sync to Y-signal
external synchronization is not possible

Data Format: 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

**Image
Resolution:** maximum 752 x 582 pixels (PAL)
or 640 x 480 pixels (NTSC)
Resolution is freely scalable in X and Y directions
up to 14:1

**Image Transfer
Rate:** Half frame 20 ms
Full frame 40 ms
Image transfer to the main memory in real time
(Bus master transfer)

**Used
Resources:** 4 kByte main memory (register field)
/INTA

Image control: Gamma correction (selectable)
Brightness (+/- 50 %)
Contrast (0 % ... 235 %)
Color saturation (U: 0...201 %, V: 0...283 %)
Hue (+/- 90°, only with NTSC)

Image Storage: 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: 12-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 uA
Input High Current	I_{IH}	-	70 uA

1 I/O Port (driven transistor, 28 V/0.8 A_{max})

Parameter	Symbol	Min	Max
Input High Voltage	V_{IH}	2,0 V	28 V
Input Low Voltage	V_{IL}	-0,5 V	0,5 V
Output High Voltage	V_{OH}	5 V	25 V
Output Low Voltage	V_{OL}	0 V	1,4 V
Input Low Current	I_{IL}	-	-700 uA
Input High Current	I_{IH}	-	70 uA
Output HiZ Current	I_{OZ}	-	500 uA
Output On Current	I_{OON}	-	800 mA
Switching frequency	f_{IO}		200 Hz

1 I^2C interface (Master)

Parameter	Symbol	Min	Max
Transmission rate [1]	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 uA
Input Low Current	I_{IL}	-	-10 uA
Output Low Voltage	V_{OL}	-	0,4 V

EEPROM: 256 Byte internal EEPROM
Write cycles: more than 10.000

---

[1] :   Both of the frequencies can be de-lactivated with software

---

<u>Supply Voltage:</u>
(only models EPC-032-X / EPC-032-X1-X2)

Input Voltage:	8 … 28 VDC

Output Voltages: +5 VDC / 2 A
+12 VDV / 1A
When using the +12 V output, the input voltage must lie within the range of 15…28 V

Output Level: 22 W max.

Shut down: electronic, TTL-Signal

<u>Connectors:</u>   *2 SMB-Sockets:*

- X1 = Composite Channel 1
- X2 = Composite Channel 2

26-pin male connector
- Composite channel 1-3
  (EPC-032-X1: channels 1-9)
- S-Video chroma signal
  (luma on Composite 9)
- 1 I²C interface
- 1 supply voltage output 1.5 A max
- 1 I/0 connector, transistor operated

Mini-DIN socket X4:
- 1 S-Video input

Male Connector 2x10 X6:
- GPIO-Port, 12 x TTL I/O
- 1 I²C interface
- 1 I/O connector, transistor operated

Power Adapter Socket X300:

▪ PC-Floppy socket connector (3.5") for feeding in a camera supply voltage. The voltage is looped through at X3

(not X2 models, see next paragraph)

Models EPC-032 and EPC-032-X1-X2 supplemental:

Supply Voltage Socket X300:

▪ PC-Floppy socket connector (3.5") for connection of supply voltage

(8 – 28 VDC, when using the 12 V output 15 - 28 V)

3 pin removable screw clamp X302:

▪ Supply voltage output +5 V/ 2 A, +12 V/ 1 A

8 pin male connector X303 (optional):

▪ Supply voltage for Jumptec-MOPS CPU-card (removable)

Socket Connector X301 (optional):

▪ Shutdown connector for shutting down the supply voltage

1.3 Field of Applications and Safety Regulations

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

- The eGrabber-4plus 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 (channel 9 only).
- 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 re-oufficient to handle the volume of the image data, otherwise information might be lost.

- The eGrabber-4*plus* is intended for use in PC/104-plus systems. The eGrabber-4plus has to be plugged into a master PCI-slot. When assembling the PC/104-plus system and installing the housing, applicable CE-norms must be taken into consideration.
- 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.

1.4 Addresses and Resources

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

Only the position of the card in the PC/104-plus system has to be set with the help of Jumper JP201 (*see section 2.1*). Up to three eGrabber-4*plus* can be installed in a single system. Several eGrabber-4plus 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 eGrabber-4plus 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.

Note:
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 CPU-motherboard. Usually the addresses are allocated in sequence of the numbering of the PCI-slots. This might deviate for different manufacturers.

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

Since the Grabber is only a *single function device* 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.

Caution:
Since several boards can trigger the same interrupt /INTA, it must be determined which board caused the interrupt.

❍ When the PC is switched on the PCI-BIOS will recognize the new card. This is shown by a corresponding report.
❍ For function control we recommend the demo-software on the included CD.

© PHYTEC Meßtechnik GmbH 2004   L-608e_4

1.5 Socket Pinout

Note:
The following description of the Grabber's connectors is intended only as a technical reference.
A detailed description of the start-up of corresponding connections and connector cables can be found in *section 2*.

The contents of *section 1.5* are not relevant during the initial start-up.

❶ = present on all modules ◎ = only X2 models
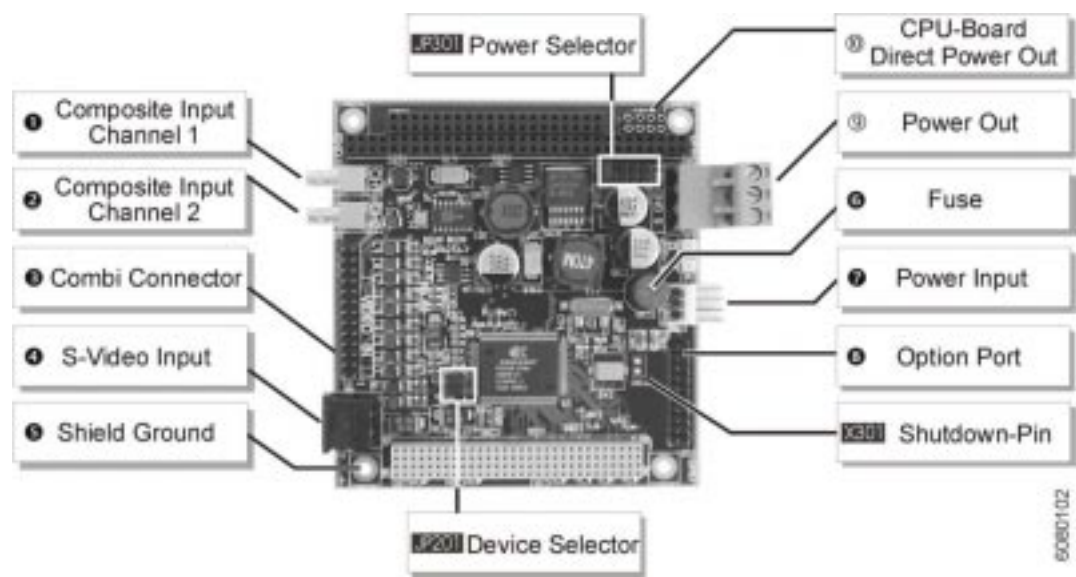
Figure 2: eGrabber-4plus Connectors

1.5.1 Composite Inputs

All composite video sources with an output level of $1\,V_{ss}$ and an impedance of 75 Ω can be used. For more information on video standards, please refer to *section 1.2*

- **Version EPC-032, EPC-032-X2**

  The Grabber has three composite inputs. Channels 1 and 2 extend to the SMB sockets ❶ and ❷. All three channels extend out to the 26-pin header row X3 ❸.

  > **Caution:**
  > It is not possible to use both connector options simultaneously!

- **Version EPC-032-X1, EPC-032-X1-X2**
  The Grabber makes available nine composite inputs. Channels 1 and 2 extend to the SMB sockets ❶ and ❷. All nine channels extend out to the 26-pin header connector X3 ❸.

  > **Caution:**
  > It is not possible to use both connector options simultaneously!

  In addition to the composite inputs there is also a supply voltage from which the cameras can be supplied with power, as long as the external supply voltage is connected to socket X300 ❼ (*see section 1.9*).

Additional signals are also available on the combi-connector X3 ❸.

eGrabber-4plus with 3 Composite Inputs
(EPC-032, EPC-032-X2)

SMB-Sockets		Pin Header Row X3 ❸	
Socket	**Function**	**Pin**	**Function**
X1 ❶	Composite 1	**1**	Composite 1
X2 ❷	Composite 2	**2**	Signal Ground
		3	Composite 2
		4	Signal Ground
		5	Composite 3
		6	Signal Ground
		7	
		8	
		9	
		10	
		11	
		12	
		13	
		14	
		15	
		16	
		17	S-Video LUMA
		18	
		19	S-Video CHROMA
		20	Signal Ground
		21	I^2C-Bus: SDA
		22	I^2C-Bus: SCL
		23	Signal Ground
		24	Signal Ground
		25	Pwr. Supply out
		26	I/O-Pin

Table 2:   Pin Assignments of the SMB-Sockets and Pin Header Rows,
*   Model EPC-032, EPC-032-X2*

**eGrabber-4plus with 9 Composite Inputs
(EPC-032-X1, EPC-032-X1-X2)**

SMB-Sockets		Pin Header Row X3 ❸	
Socket	**Function**	**Pin**	**Function**
X1 ❶	Composite 1	**1**	Composite 1
X2 ❷	Composite 2	**2**	Signal Ground
		3	Composite 2
		4	Signal Ground
		5	Composite 3
		6	Signal Ground
		7	Composite 4
		8	Signal Ground
		9	Composite 5
		10	Signal Ground
		11	Composite 6
		12	Signal Ground
		13	Composite 7
		14	Signal Ground
		15	Composite 8
		16	Signal Ground
		17	Composite 9 and S-Video LUMA
		18	Signal Ground
		19	S-Video CHROMA
		20	Signal Ground
		21	I^2C-Bus: SDA
		22	I^2C-Bus: SCL
		23	Signal Ground
		24	Signal Ground
		25	Pwr. Supply out
		26	I/O-Pin

*Table 3:   Pin Assignments of the SMB-Sockets and Pin Header Rows,
           Models EPC-032-X1, EPC-032-X1-X2*

PHYTEC offers connector cables among its included accessories, which enable signal connection to the SMB-sockets ❶ and ❷. The WK023 cable (*see section 1*) fits on both SMB-sockets (video inputs 1 and 2).

1.5.2 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.

There are two possibilities for connecting an S-Video source to the eGrabber-4plus:

- It is possible to direct an S-Video signal to the 4-pin Mini DIN socket (X4,❹). The socket is switched to the corresponding S-Video norms. The connection of the camera is possible using an S-Video cable.

- An S-Video camera can be connected the 26-pin socket connector (X3, ❸). The S-Video chromatic signal is connected to Pin 19. One of the composite inputs is used to feed in the S-Video luma signal (brightness)(for models with 9 channels this is channel 9, for models with 3 channels it is channel 3).
  On all models this signal extends to Pin 17 on X3.

  If you choose to procure your own corresponding connector cable the following pins must be connected (connecting the supply voltage (Pin 25) is optional, for this voltage must be connected to socket X300):

26-Pin Header Row X3 ❸	
Pin	**Function**
17	S-Video: Luma
18	Signal Ground
19	S-Video: Chroma
20	Signal Ground
24	GND
25	Spg. out (Kamera Spg.)

Table 4:   Connecting the S-Video Input to Pin Header Row X3

Caution:
Both S-Video inputs can **not** be connected at the same time. Either the Mini DIN input ❹ **or** the connection on the 26-pin header row X3 ❸ can be used.
In the user program the selection of the socket can occur automatically.

1.6 I/O Pin

On the pin header row X3 as well as on the Option Port - pin header row X6 (*also see section 1.8*) there is an I/O-Pin available for universal use.

This pin can be used as an input as well as an output. The input can be called from the program and is not connected to any special functions.

When using the function as output, the user program is able to convey control signals to other devices. The Output can be set, or cleared by the program.

I/O Pin Functions

- **Input**
  An external voltage (against Ground) can be connected to the I/O pin. If the voltage is between 0 V and 5 V, then the program receives a „0". If the voltage is between 2 V and 30 V, then the logic signal level is set to „1". The positive connection must be at pin 26 (X3) or at Pin 15 (X6).

Pin Header Row X3		Pin Header Row X6	
Pin	**Function**	**Pin**	**Function**
24	Ground	19	Ground
26	I/O-Pin (+)	15	I/O-Pin (+)

Table 5:   I/O-Pin Connections on Pin Header Rows X3 and X6
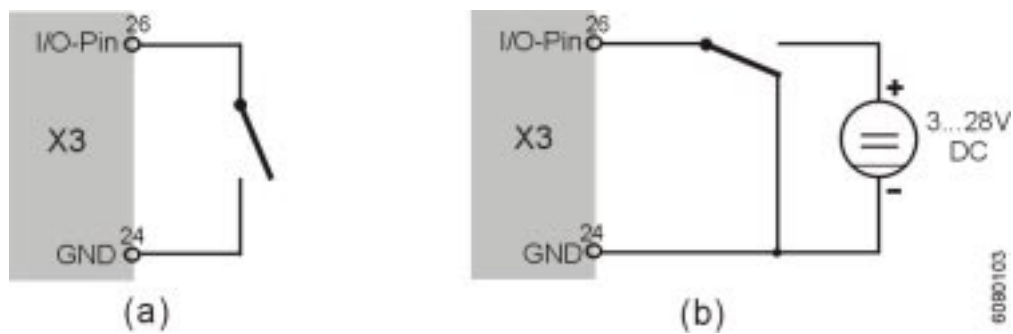
Figure 3:   Standard Connections for the I/O Pin as Input

- **Output**

  For use as an output pin, Pin 26 or Pin 15 is connected to Ground through a circuitry transistor (for example Pin 24 for X3, Pin 19 for X6), if the program sets the Pin to a logical „1".

  If the program sets the pin to logic „0", then the transistor is disabled and thus, there is no connection between pin 26 or Pin 15 and Ground.

  In order to use the output, an external power supply, in the range of 5 V and 28 V, is required. It is also possible to use the power supply pin (pin 25 of X3) for power supply.

  Figure 3 and *Figure 4* depict two possible connections.
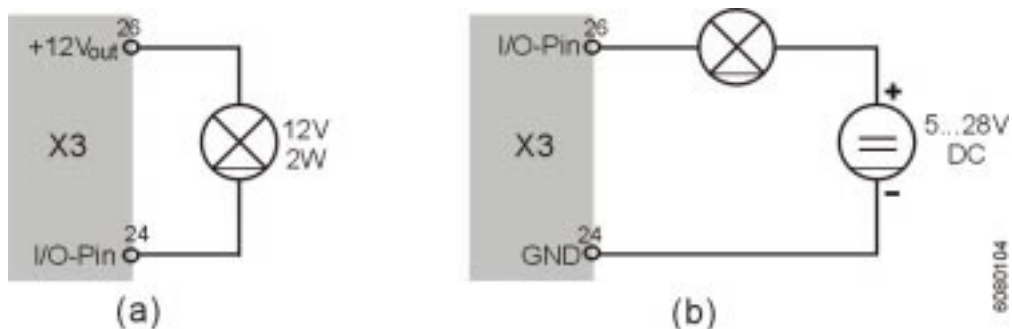
a)Connection from the Grabbers' power supply pin
b) Connection from external DC voltage supply source

Figure 4:   Standard Connections for the I/O Pin as Output

Caution:
The polarity of the connected voltage must be selected so that the I/O pin 26 (X3) or 15 (X6) has a constant positive potential.
Negative potential (i.e. Ground) at pin 26 or 15 can lead to destruction of the Grabber card!
While using the output function (I/O-Pin to Ground), the operating voltage must be between +5 V and +28 V. When using the I/O pin as input, the operating voltage must not exceed 28 V.
The I/O-Pin is not galvanically isolated from the video lines, the CPU or the remaining signal lines.

1.7 I²C Interface

External devices can be polled or controlled via the I²C interface. In order for this to occur, the external devices must have an I²C interface operating in Slave mode.
The I²C interface is available at pin header row X3, and is also available on the internal pin header row of the *Option Port* X6 (*also see section 1.8*). It is possible to connect multiple I²C devices to the bus, but these devices must be distinguished by their device addresses.

Table 6 shows the relevant pin assignment of the pin header rows.

Pin Header Row X3		Pin Header Row X6	
Pin	**Function**	**Pin**	**Function**
21	I²C Bus: SDA	18	I²C Bus: SDA
22	I²C Bus: SCL	17	I²C Bus: SCL
23	Ground	19	Ground

Table 6:   Connecting the I²C interface to Pin Header Rows X3, X6

Note:
The maximum cable length is restricted, due to the fact that the I²C interface is driven by TTL signals. For a connected device, depending on the configured transmission rate, the maximum cable length is aprox. 1 - 2 m.
Use cables with sufficient shielding when connecting this device.

Information for adapting the I²C interface into user software can be found in *section 5.2.7,* under the functions group „*Transmitting Data via the I²C Interface*".

Note:
The device address range 0xA0 to 0xA3 is used internally on the Grabber card. For this reason it is not possible to configure an external I²C-device on these device addresses.

1.8 Option Port

The option port provides 12 digital I/O-lines and one I²C interface to the user. The signals are routed to a connector with 10 x 2 pins. The connector is denoted as X6 (❽). *Figure 5* shows the assignment of the pins.

Note: The current drawn out of pin 1 (+5V) may not exceed 100 mA.
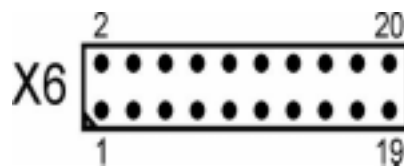
Figure 5:   Pin Formation of the Option Port

Option Port, X6								
pin	**function**		**pin**	**function**		**pin**	**function**	
1	+5V out		8	I/O 6		15	I/O-Pin	
2	I/O0		9	I/O 7		16	I/O Clk	
3	I/O1		10	I/O 8		17	I²C SCL	
4	I/O2		11	I/O 9		18	I²C SDA	
5	I/O3		12	I/O 10		19	GND	
6	I/O4		13	I/O 11		20	GND	
7	I/O5		14	N.C.				

Table 7:   Pin Assignment for the Option Port

Note:
The I/O-Pin (X6 – 15) is identical to the I/O-Pin on the pin header row X3 (X3 – 26). Therefore only one of these pins can be connected. *Additional information about the I/O-Pin can be found in section 1.6.*

1.9 Power Supply Output

A supply voltage output is available on pin header row X3, enabling voltage supply to external components such as a camera (*shown in Table 8*).
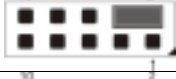
The output currant may not exceed 1.5 A. The output voltage may not be negative.

Pin Header Row X3	
Pin	**Function**
24	Ground
25	Supply Voltage (+)

Table 8:   Supply Voltage Output

With the removable jumper JP301 you can select between different supply voltage options. Please note that some options are only possible with certain Grabber types.

Overview of Options

		Model	
Supplying X3-25:	Jumper JP301	EPC-032 and EPC-032-X1	EPC-032 and EPC-032-X2
Input Voltage (X300)		●	●
+12 V from PCI-Bus		●	○
+5 V from PCI-Bus		●	
+ 12 V form integr. adapter			●
+ 5 V from integr. adapter			●

● = possible ○ = possible, however not recommended

Table 9:   Options for the Supply Voltage Output (X3, Pin 25)

Description of Options:

Note:
Pin 1 of the jumper field is identified by an angled corner on the stencil.

- Power supply from the input voltage

Figure 6:   Jumper Setting JP301

The input voltage (X300 – Pin 4) is connected to Pin X3-25. The voltage is unregulated. In the power circuitry there is a diode for protection against reverse polarity along with a circuit breaker SI301 (2 A).

Caution:
In this configuration the voltage at Pin 25 corresponds to the voltage of the power supply device. Take this into consideration when connecting external units to this pin.
Under no circumstances should the voltage connected to socket X300 exceed +28 V. The current draw must not exceed 1.5 A. The output is protected by the circuit breaker F1 (replacement circuit breakers can be provided by PHYTEC, order number KF012).

- Power supply from the PCI-Bus with +12 V
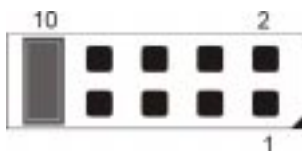
Figure 7:   Jumper Setting JP301

The +12 V voltage, which is carried over the PC/104-plus connector to the Grabber card is connected to Pin X3-25.

This setting is not recommended if you have a Grabber card with an integrated supply voltage (versions EPC-032-X2 / EPC-032-X1-X2).

Please note that the current load capacity of the PCI-Bus is limited. The expressed maximum current draw of 500 mA should not be exceeded.

Note:

This option functions only if the PC/104-System is connected to a supply voltage of +12 V.

- Power supply from the PCI-Bus with +5 V
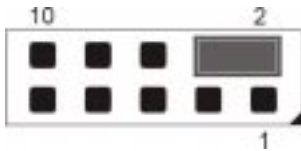
Figure 8:   Jumper Setting JP301

The +5 V voltage, which is carried over the PC/104-plus plug to the Grabber card, is connected to Pins X3-25.

This setting is not a option if you have a Grabber card with an integrated power supply (versions EPC-032-X2 / EPC-032-X1-X2).

Please note that the current capacity of the PCI-Bus is limited. Current draw should never exceed 500 mA.

- Power supply from the integrated power supply with +12 V

- **versions EPC-032-X2 / EPC-032-X1-X2 only**
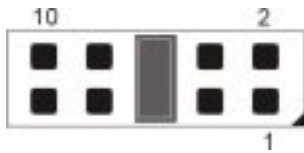
Figure 9:   Jumper Setting JP301

- A regulated +12 V from the internal supply is connected to pins X3-25.

> **Note:**
> This option is only available if you have a Grabber version with an integrated power supply and the input voltage (at X300) amounts to more than 15 VDC.

- Power supply from the integrated power supply with +5 V

  – **versions EPC-032-X2 / EPC-032-X1-X2 version**

Figure 10:   Jumper Setting JP301

A regulated voltage of +5 V from the internal supply is connected to pins X3-25.

> **Note:**
> This option is only available if you have a Grabber version with an integrated power supply.
> The maximum current draw from the 5 V branch of the integrated supply voltage amounts to 2 A.

1.10 Integrated Power Supply (only for X2-Models)

The eGrabber-4plus is equipped with a power supply in model versions EPC-032-X2 and EPC-032-X1-X2.

The integrated power supply allows for the operation of the Grabber and other components of the PC/104-System with an unregulated direct current, for example from a car battery.

In many applications this renders the insertion of a special voltage supply card unnecessary.

The eGrabber-4*plus* can make two output voltages available:

- +5 V, max 2 A
- +12 V, max 1A

Note:
The power supply is only present on the following models:

● EPC-032-X2
● EPC-032-X1-X2

1.10.1 Connecting the Input Voltage

An unregulated direct current can be fed into the supply voltage.

Allowed input voltage range:

Applied Output Voltages		Input Voltage Range
+5 V	+12 V	
●		8 V... 28 VDC
●	●	15 V... 28 VDC

Table 10:   Allowed Input Voltages EPC-032-X2

Caution:
The input voltage must not exceed the prescribed maximum value. For input voltages with high levels of interference, especially if power spikes occur, an additional interference filter is required.

The input voltage is connected to the socked connector X300 ❼. The socked connector corresponds to the supply voltage connector from the 3.5" floppy disk drives. *Figure 11* shows the pin-out of the supply voltage input.
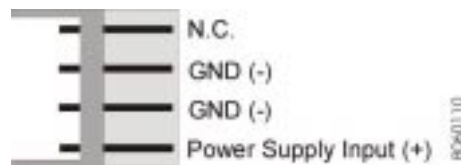
Figure 11:   Connecting the Supply Voltage Input X300

The Grabber is equipped with a diode for protection against polarity reversal.

The maximum allowed input power is 22 W. The power supply is protected by the circuit breaker SI301 ❻. The maximum allowed input current is 2 A.

1.10.2 Output Supply Voltage

The regulated output voltage is available at socket X302 (⑨).
Figure 12 shows the socket's pin-out.

Figure 12:   Connecting the Supply Voltage Output X302

Connect the output X302 with the supply voltage input of the
CPU-card you inserted. It is important to note the pin-out of the
CPU-card.

Note:
- The output voltage is not directly connected to the PC/104-Bus
  pins. Since CPU-cards tend to require greater current, the supply
  for the PC/104-System is fed over the CPU-card and from there
  distributed over the bus.
- The Framegrabber is not fed directly by the supply voltage, rather
  over the bus connector of the PC/104-System.
- It is also possible to connect peripheral devices (floppy disc drives,
  hard drives etc.) to socket X302. For hard drives and similar
  components it is important to be careful that the beginning start-up
  current during a power up does not exceed the maximum allowed
  output current. In this case the supplies surge protector takes effect
  and the components won't start correctly.
- The presence of the operating voltage is shown by the LEDs D307
  / D308 next to the socket X302. If an LED does not illuminate, or
  illuminates only slightly, then it is possible that the integrated
  surge protector on the adapter was activated. When the current load
  is reduced the supply will once again deliver the correct output
  voltage.

CPU-cards from the MOPS-series by Jumptec can be connected directly with the power supply of the eGrabber-4*plus*. To facilitate this, socket X303 (⑩) is located at the same poition as the voltage input of the MOPS-CPU-cards.

By using appropriate pin connectors, the supply voltage can be delivered without connector cables, simply by linking the cards together.

The socket X303 is designed especially to fit the aforementioned CPU-card and delivers all required voltages.
For additional information refer to the handbook of the MOPS-CPU card

Important:
Test exactly whether the pin-out of the supply voltage connector on your CPU-card matches the pinout of socket X303 (see *Table 11*).

Pin	Function
1	GND
2	+5 V
3	N.C.
4	+12 V
5	N.C.
6	N.C.
7	GND
8	+5 V

Table 11:   Pinout of Socket X303

1.10.3 Supply Voltage Options

- Supply over PCI-Bus
  The output voltage of the adapter can be carried directly on the PCI-Bus. The cable connection between the Grabber and the CPU-card is then no longer necessary.
  This option is generally not recommended, since the connection via a specially configured socket is optimized by the CPU-cards manufacturers.

  JP 301 must be jumpered as follows in order to connect the adapter output with the corresponding pins of the PC/104-Bus connector:
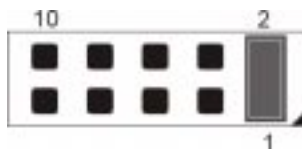
Figure 13:   JP301: 5 V on PC/104-Bus Connector
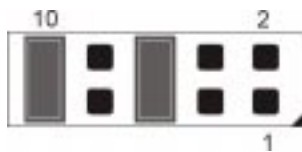
Figure 14:   JP301: 12 V on PC/104-Bus Connector

Note:
Connection of the +12 V output with the PC/104-Bus connector is only possible if the +12 V output voltage is also connected to the supply voltage output X3-25.

- Power down power supply (Power-Down Pin)

  The integrated adapter can be shut down electronically. This can be accomplished via a switch on the connector field X301 or with a TTL-signal.

Figure 15:   Connector Field X301

The power supply is shut off when X301-Pin1 is connected to GND.
In addition to this, both of the X301 pins can be connected to one another, for example via a switch.
A controller signal can be connected to X301-1 as well, over which an adapter can be switched on and off.

1.11   Device Selector

With PC/104-Systems, the position of an insert card in the PCI-Bus system has to be set using a Jumper. One can imagine the position as the card's logical position in the PCI-System or as a „device address". The position is also defined as „Slot", „Slot-Number, or sometimes as „Device-ID".

No two cards in a PCI-System are allowed to be set at the same position. The system will most often not boot at all in such cases.

As a rule, PC/104plus-CPU-Cards have up to four positions available.

- Find out which positions your CPU-Card makes available. The position required for the eGrabber-4*plus* must have master capability.

- Check which positions are used by other PCI-Expansion Cards in your system. Look for a free, master capable position for your card.

Figure 16:   Jumper Settings of the Device-Selector JP201

1.12 Notes on CE-Conformance and Immunity Against Interference

When building the PC/104-System, be sure to adhere to regulations regarding CE-Conformity and interference protection.

Since PC-systems run with high clock frequencies, there is a risk of interference from and through electromagnetic fields. This can be countered by careful construction of the system.

Pay close attention to the Ground potential routing within the system. High frequency Ground interference can be radiated across the video cable's outer shielding. It is a good idea to connect the shielding potential with a Ground potential (housing or PE-connector), near the input socket in this case

The eGrabber-4*plus* offers various Ground connection possibilities for adaptation to different operating conditions. Three of the four connector sockets are in contact with GND. The GND potential of the Grabber card can be connected with the Ground of the CPU-card, the housing or PE using a cable connection or conducting bolts. By using synthetic conducting bolts it is possible to isolate the Ground potential. This may be desirable in the event that you want to avoid a Ground loop.

The Shield-GND connector ❺ located next to the S-Video socket establishes a connection with the shielded connector of the S-Video socket.

In order to prevent radiation interference over the Ground potential routing, it is generally a good idea to send the connector line through a ferret. The ferret should be positioned flush with the Grabber and the cable should be run through it one or two times (creating a loop) depending on the dampening effect required..

For video lines a clamp ferret will work suitably.

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

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

2 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 in the PC/10plus insert card series, please take the time to familiarize yourself with the instructions and equipment. The following tasks are not difficult, but must be done with caution.

2.1 Installing the Grabber Card

Caution:
The PC/104plus-System must be isolated from the supply voltage.
Please ensure that the device does not have any power supplied to it.

The specifications instruct you to install the card on the virtual positions (slots or Device-ID) 1-4. Select one of these positions, be sure, however, that it is a position with master capability (in some systems not all systems are master capable).

- Two cards cannot have the same position (for example, two cards use position 3)

- Once a position has been selected, the selection must be set using Jumper JP201 prior to installation (s*ee Figure 17*).

Figure 17: Jumper Settings JP201 (Slot Selection)

> **Note:**
> Refer to the CPU-board manufacturer's manual if you are not sure whether the position in question is a Master-Position.

> **Caution:**
> If the eGrabber-4plus is installed into a Slave slot, it is possible that the system will no longer start-up (boot). In any case, the eGrabber-4plus will not function correctly. For some systems only *one* Master at a time can be installed at position 3 or 4.

- Carefully mount the eGrabber-4*plus* on the PC/104 and PC/104-plus connector row of the main board. The Grabber can also be connected to other cards in the system, you only have to make sure that all relevant CPU-card signals are available.
- The Grabber card can only be correctly connected and inserted when oriented properly.
- 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 slot Lign up the golden contact strips with the PCI slot's receptacle. Some resistance will be encountered as the contact strips spreads apart the contact springs.
- Make sure that no mechanical or electrical contact exists between components on neighboring cards. Especially due to the construction of the PC/104-cards, it is possible that components with a high profile can come into contact with one another when inserting the card, or even that full insertion of the card is rendered impossible because of this. Therefore, whenever possible, select a different arrangement for the cards or employ a spacing insert to maintain the proper distance between the cards.
- Secure the card to the assembly holes with the aid of screws and spacing pins.
- 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 in place with the four screw holes and bolts.

2.2 Power Supply

Generally the PC/104-System is supplied with power over the CPU-card. Check, based on the CPU-card instructions, which voltages are required and how they are to be connected.
The Grabber requires a supply voltage of +5 V. The CPU-card usually directs this voltage to the Grabber card over the PCI-Bus connector. Therefore no special power supply board is required.

- The power supply for the camera at pin header row X3 can also occur over the Framegrabber card. Additional information on this subject can be found in *section 1.9*.

  The Grabber models **EPC-032-X2** and **EPC-032-X1-X2** each have a separate supply voltage adapter, which is capable of supplying the entire PC/104-System with power.
  The power supply is completely independent from the Grabber.
  In order to use the integrated power supply, connect a DC voltage in the range of 15 to 28 VDC to the input connector X300.

  If you do not use the +12 V output for the supply voltage, the input voltage range can be from 8 to 28 VDC. The pin-out of X300 is shown in *Figure 18*.
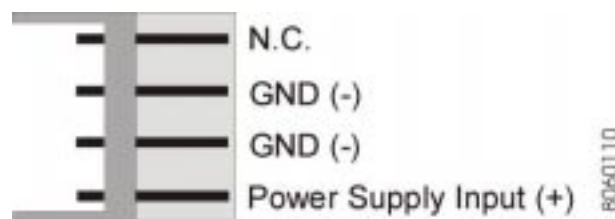
Figure 18:   Pin Out of the Supply Voltage Input

The regulated output voltages +5 V and +12 V are connected to the screw clamps on X302 (see *Figure 19*). You can connect the CPU-card's supply voltage inputs there. With the two green LEDs you can check the output voltage. Be sure that the output voltages are not redirected to the Grabber or distributed over the PC/104-connector. Therefore all components that are to be supplied with voltage have to be connected to X302 in standard configuration. *Additional information on the integrated supply voltage is available in section 1.9.*

Figure 19:   Supply Voltage Output

Note:
The voltage to socket X300 must not exceed 28 V.
The current draw is limited to 1.5 A by the circuit breaker F1.
(Circuit breakers are available through PHYTEC under order number: KF012)

2.3 Installing the Driver

In this step, the softwaredriver and demo program is installed.
This may be different for each driver, depending on the operating system. How this works is explained below.

- Switch on the PC/104plus-System.
  During initialization the computer's BIOS should recognize the card automatically. Many BIOS versions will briefly show a list of the located PCI-devices. The eGrabber-4*plus* appears twice in this list as „Multimedia Device".

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' 95^{TM}:**

After the operating system has recognized the card, a window appears, *New Hardware Component Found,* offering the user to install the driver. From this window, select the „*different position*" option and confirm with *OK*.

A new window will appear entitled „*select different position*". Place the **PHYTEC Vision Utilities** CD into the CD-ROM. Choose *search* and in the window that will appear, select the CD-ROM drive.

Change the path to ***pciGrab4\driver\win95_98.*** Confirm by selecting *OK*.

A list appears on the CD, which names the found drivers. Select *PHYTEC PCI-Grabber* from the list. Now the driver should be automatically installed from the CD to the computer.

In the window that will appear next, confirm a Restart of the computer. After start-up the computer should properly function with the operating system.

The driver has now been successfully installed.

Please *refer now to section 2.3.1.* Then *refer to section 2.4,* which explains how to install the demo software.

- **Windows' 98/ME/2000/XP^{TM} :**

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 **pciGrab4\driver\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 2.3.1*. Then *refer to section 2.4* to find information on how to install the demo software.

- **Windows' NT4.0^{TM} (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' 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 2.3.1.* Then *refer to section 2.4* for information on how to install the demo software.

2.3.1 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.*

The **„LabView" driver** works with the measurement and automation programs from National Instruments (IMAQ package is required). For more information on the program or the driver, please *refer to the LabView's User's Manual.*

If installation of these drivers are 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 *PCI-Grabber*. An installation window will appear next containing the following two entries:

- *Install Twain*
- *Install LabView*

2.4 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 installation menu that will appear (*see Figure 20*).
- Click on *Install Windows demo software.*
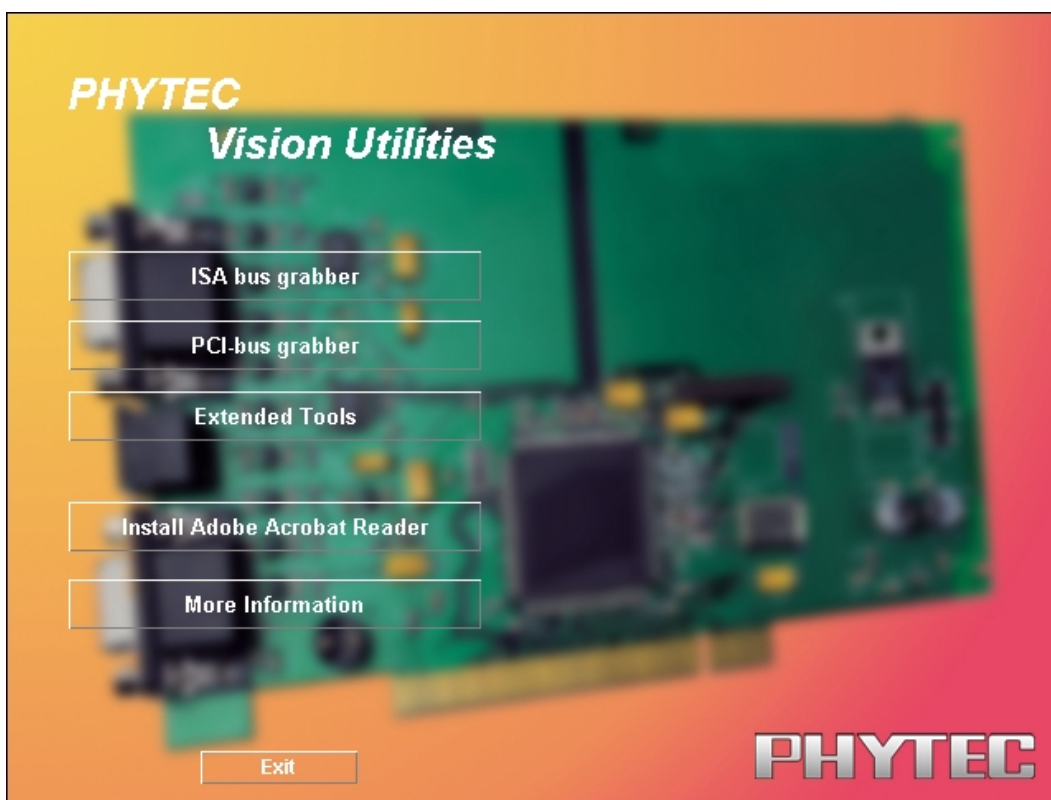
Figure 20:   PHYTEC Installation Menu

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

© PHYTEC Meßtechnik GmbH 2004   L-608e_4

3 Connecting Video Sources

It is possible to connect one or more video sources to the eGrabber-4plus (*see Figure 21*). 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, both 3 composite and one S-Video (*EPC-032-X1, EPC-032-X1-X2*), or 9 composite and one S-Video source (*EPC-032, EPC-032-X2*)) can be connected to the Grabber.

Changing channels occurs via software, or via the included demo program.
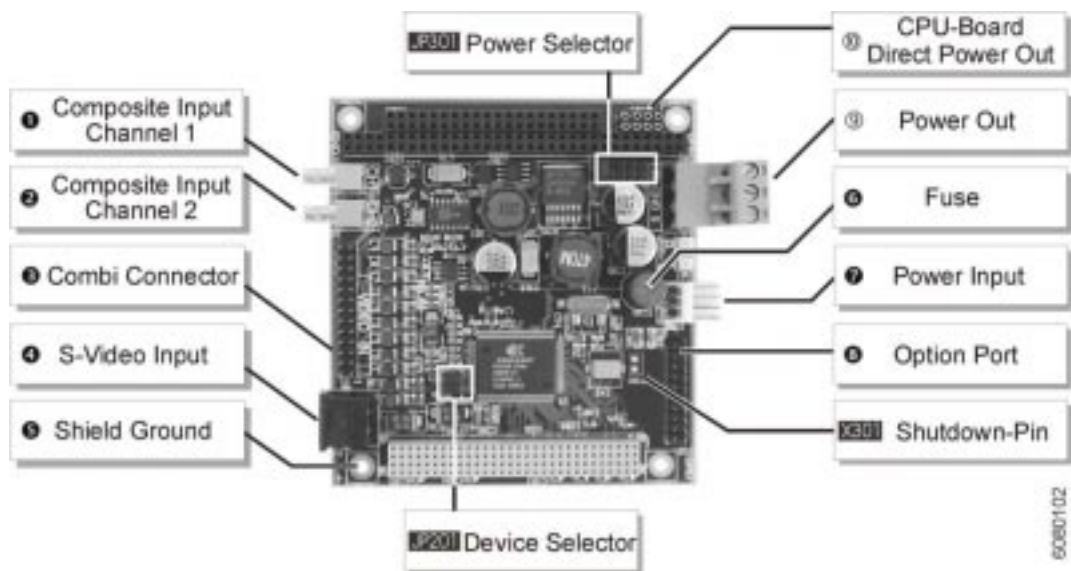
Figure 21:   Overview of the eGrabber-4plus Connectors

The composite inputs are located on the SMB-Sockets ❶/❷ and, on the 26-pin header row X3, ❸. A camera can be connected to a supply voltage via X3 as well.

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

3.1 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 22*).
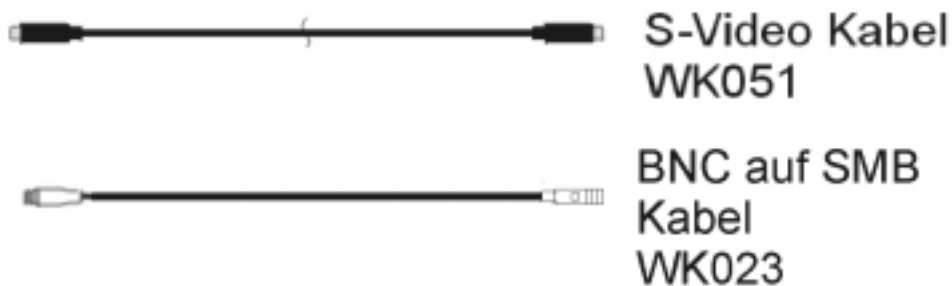
Figure 22:   Video Connector Cables - (Description and PHYTEC Order Number)

Refer to the instructions to your video source in order to check for connector compatibility.

The following section briefly describes the above depicted cables.

© PHYTEC Meßtechnik GmbH 2004   L-608e_4

3.1.1 The S-Video Cable

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

3.1.2 The Composite Connectors

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 Ω) must be used.

The end with the SMB-plug is inserted into the Grabber (socket ❶ or ❷).

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 4*).

4 Start-Up of the Grabber with the Demo Program

In order to continue with this section, the demo program and the Grabber driver must be correctly installed (*see section 2*).

The demo program can be found under ***START / Programs / Phytec / pciGrabber4plus / Grab4PCI.*** After this program has been started, an empty program window will appear with menu options (*see Figure 23*).
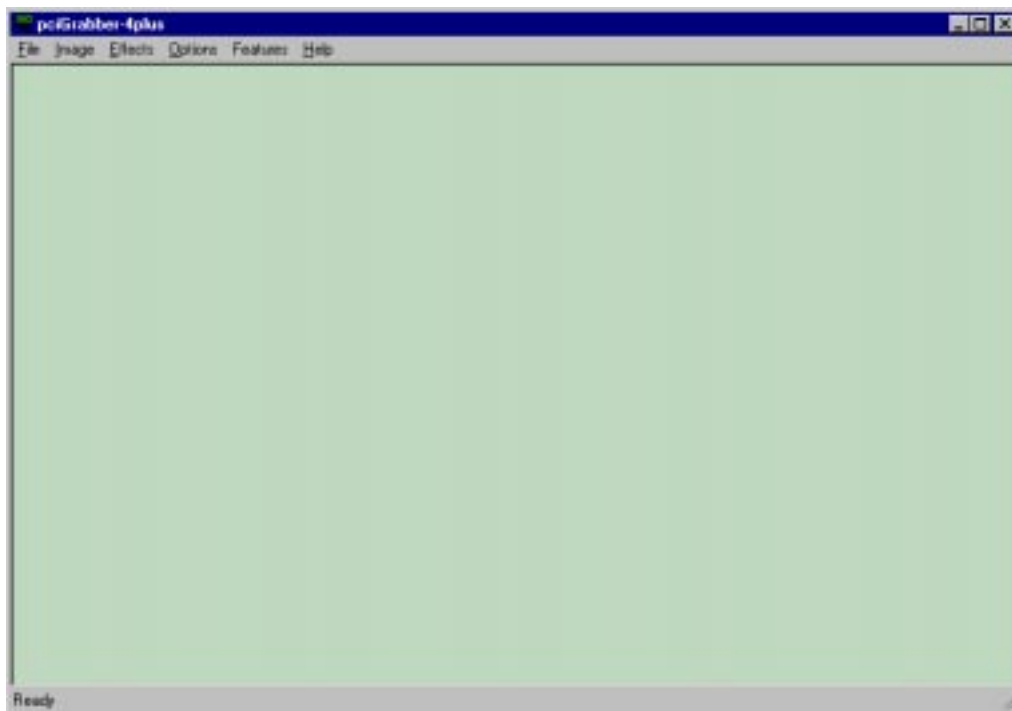
Figure 23:   Overview of the Demo Program

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.

Click on the *Image* button and the following pull-down menu will appear (*see Figure 24*).
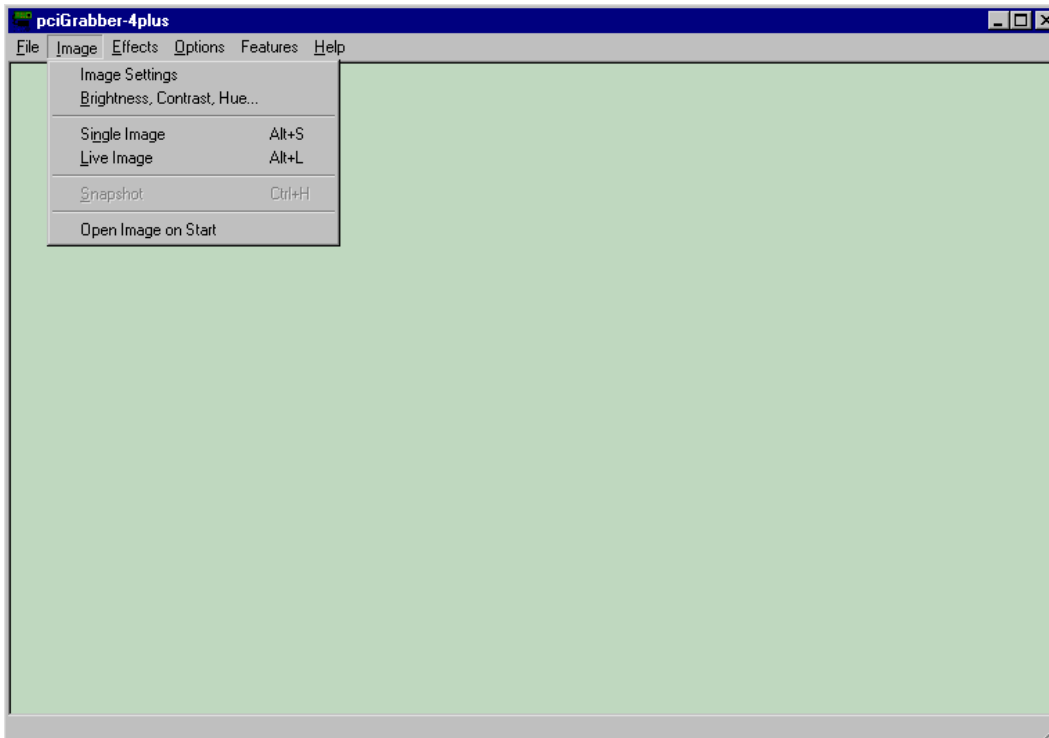
Figure 24:   Menu Option: Image

In order to configure the parameters of the image to be grabbed, select the *Image Settings* command from the pull-down menu (*see Figure 25*).
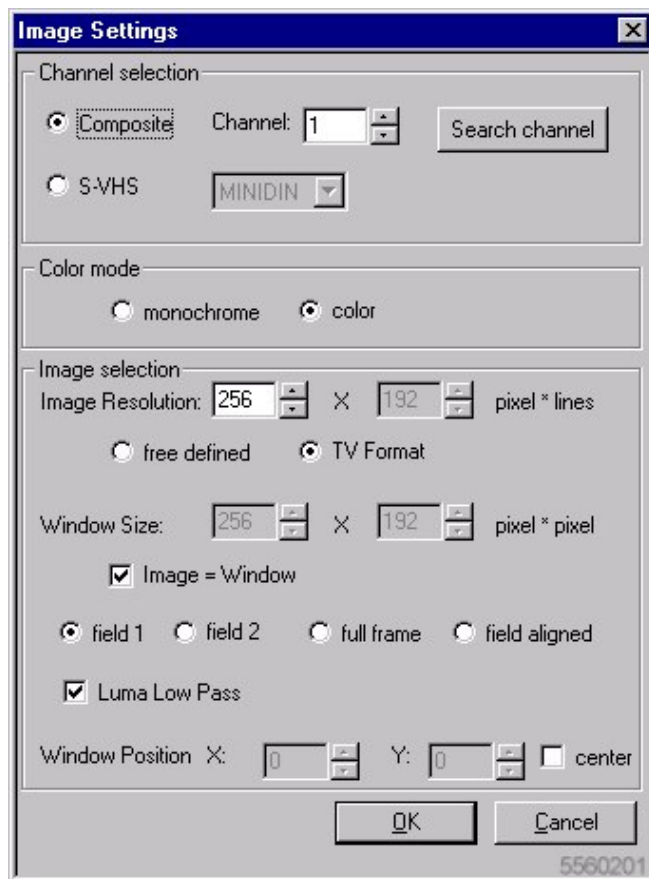
Figure 25:   Configuring the Image Parameters

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.

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 26)*
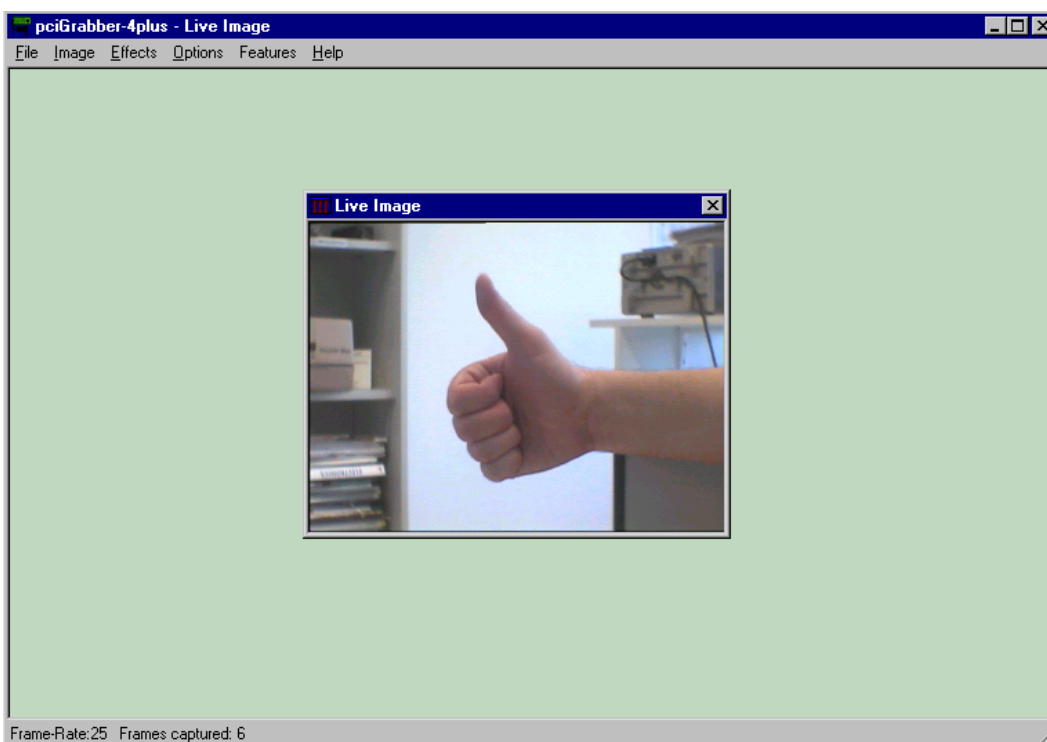
Figure 26: Live Image from the Video Source

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 sources are described in *section 7, "Trouble-Shooting"*.

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/104 systems.

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.

4.1 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 27)* contains parameters that influence image generation and depiction:
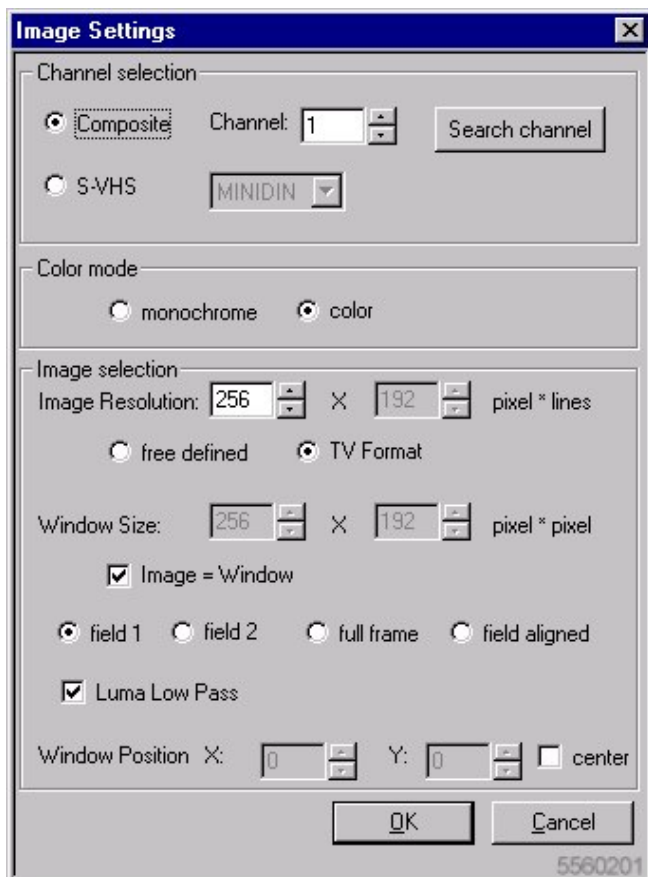
Figure 27:   „Image Setting" Menu

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 both of the SMB-Sockets ❶/❷and to the Composite over the pin header row X3
  From the *Channel* menu, select the appropriate input channel for the connected camera.
  Clicking on *Search Channel* allows the grabber to search for an active input channel. The program configures the first channel with a video signal.

- **S-Video Source**

  S-Video (or „S-VHS") – Two possibilities exist for connecting a source to the Grabber. Over the Mini-DIN plug ❹ or over the pin header row X3 ❸.
  Select S-VHS MINIDIN for the Mini-DIN plug ❹ or S-VHS COMBI for the header row X3 ❸.

  > **Note:**
  > Only **one** S-Video connection possibility can be implemented at any given time.

  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 if 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 then 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 28*). 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 28:   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 29).*

Figure 29:   Comb Effect That Occurs with Quick Moving Objects

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

With vertical resolutions lower 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 fields per second. This eliminates the 20 ms pause between the digitization of fields.

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

This occurs because the two fields cannot be interlaced to form a full frame. When configuring *field aligned*, the Grabber automatically moves the second field one half row, so that the second field can properly interlace with the first field, creating a 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 aprox. 20 ms).

If the horizontal resolution is smaller than 360 Pixel the checkbox came *Luma 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.

4.2 Image Control

During the displaying of a live image the image control window can be opened from *Image* pull-down menu under auf *Brightness, Contrast, Hue...* . the dialog shown below appears.

Figure 30:   The Image Control Window

Using the sliders it is possible to modify settings for brightness, conrast, color saturation and color tone.
The values are transferred immediately to the Grabber, so that the effects of justification can be seen and judged immediately in the live image.

The *Brightness* slider can be used to adjust brightness levels to meet environmental conditions.
The *Contast* slider can be used to adjust the image contrast as needed.

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.

© **PHYTEC Meßtechnik GmbH 2004   L-608e_4**

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. Please leave the slider here in the middle position.

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 modify the white balance at the camera or video source if possible).

4.3 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.

4.4 Crosshair function (Overlay)

Several types of crosshairs can be overlaid in the live image. This can be useful to a center a objet 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 overlaid in the image.

4.5 Basic Parameters

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

Basic Settings contains the following menu:
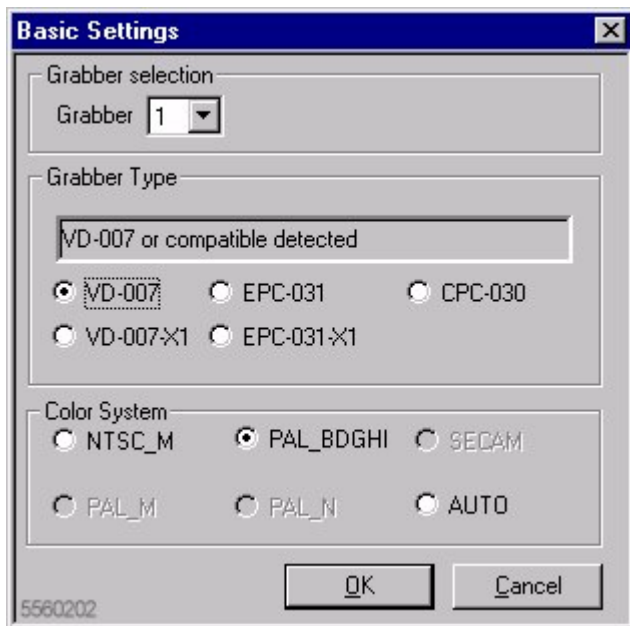
Figure 31:   Basic Settings Menu

Multiple eGrabber-4*plus* units can be operated in a PC/104-System. Select the appropriate number in the *Grabber selection* field. If there is only one Grabber installed in the computer, then the Grabber is automatically activated and assigned with the number 1.

Grabber Type displays which type of Grabber model is installed in the computer.

EPC-032, *EPC-032-X1* or *EPC-032-X2*, *EPC-032-X1-X2* (depending on Grabber model) denotes the eGrabber-4plus.

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.
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*.

4.6 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 gray- 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 32*).
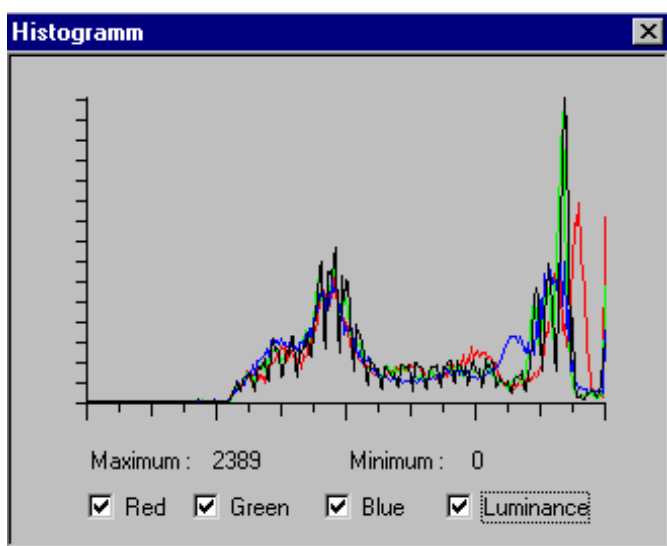
Figure 32:   Histogram

The X-axis includes the values between 0 and 255. Using the check boxes in the histogram window, the curves of gray 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 33*.
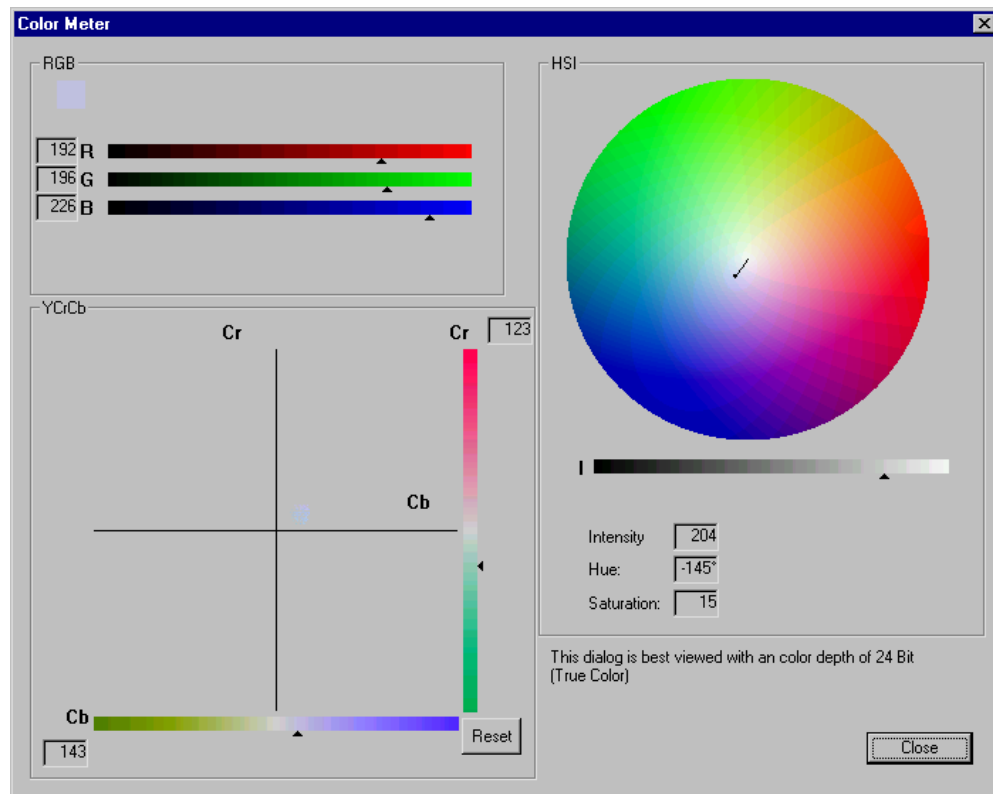
Figure 33:   Color Meter

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 aprox. 515 pixels.

- **Arithmetic Operations on Static Images:**

  The *Arithmetics* menu option provides some simple arithmetic operations on static images (*see Figure 31*).
  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).
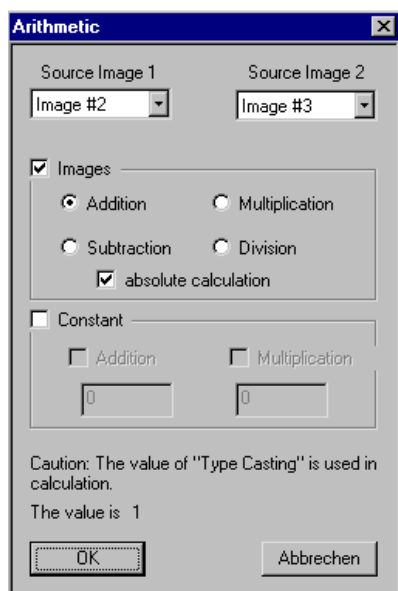
Figure 34:   Arithmetics Menu

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 35)*.
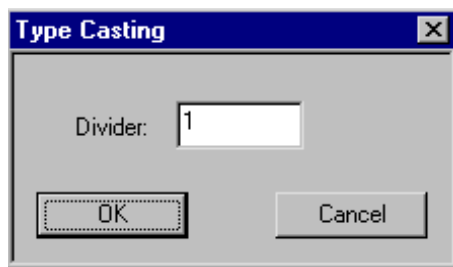
Figure 35:   Selecting the Normalization Factor

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 36*).
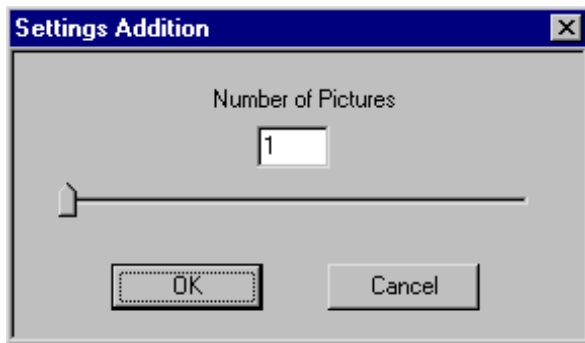
Figure 36:   Number of Images

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.

- **I/O Port Test:**
  Select the *I/O Test* command from the *Test Hardware* pull-down menu. The window shown in *Figure 37* will appear.
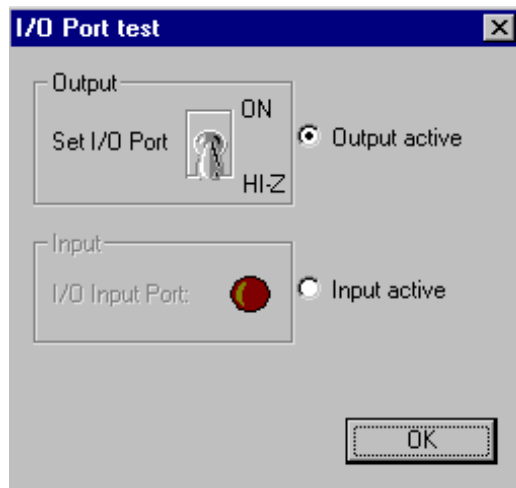
Figure 37:   I/O Test Menu

This dialog box enables the user to test the I/O port. The port is an input/output switch that is controlled by a transistor that populates the Grabber.

The port can be wired to pin header row X3 or over the Grabber's Option Port X6 (*see section 1.6*).

If *Output active* is selected, then the port pin acts as an output. The *Set I/O Port* switch can be used to activate the output (ON = connection against Ground) or deactivate the output (HI-Z = high impedance).

Selecting *Input active* allows the user to test the *I/O Input Port.* The simulated red light indicates if the port reacts to an external signal.

If the pin remains open (unconnected), the input is activated after the button has been selected. The red light will illuminate and indicates that state.

4.7 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.

4.8 Getting Started with Linux

Using the PHYTEC-framegrabber cards ‚pciGrabber-4plus‘ and ‚eGrabber-4plus‘ with Linux is easy, because these cards are supported by the standard framegrabber-driver ‚bttv‘.

The pciGrabber-4plus has an own card definition in the bttv-driver. If your bttv-driver version does not contain this card definition, please upgrade to a newer bttv version.
The PHYTEC-cards are included in version 0.7.107 or higer. Current versions can be found on the bttv-driver page *www.bytesex.org*.

Please note that the bttv-driver does not detects the pciGrabber-4plus automatically.
The driver has to be configured by the following procedure:

Step 1: Choose the card number which corresponds to your grabber version from the list below.

Step 2: Edit the *file etc/modules.conf* using a text editor. Enter the card number as show below:

```
/etc/modules.conf:

...
alias char-major-81 videodev
alias char-major-81-0 bttv
options bttv card=106  ← enter your card number here
...
```

List of card numbers		
Card No.	**Grabber Model***	**S-Video-Input**
106	EPC-032	Mini-DIN-plug
107	EPC-032	Combi-connector
108	EPC-032-X1	Mini-DIN-plug
109	EPC-032-X1	Combi-connector

*) and the corresponding –X2 models

Hints:

- The selection of the card number also defines, whether the Mini-DIN or the Combi-Connector is used as the s-video-input. In difference to the driver for Microsoft Windows, this selection cannot be done during runtime but only by the card number selection. This is, because the bbtv-driver does not support more than one s-video-input.
- For testing the correct function of the driver, we suggest to use the XawTV application.

Part 2

Programmer's Manual

5 Driver Software

This section gives you the information how you can access the eGrabber-4plus 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' 95/98/ME/XP
● Windows' NT 4.0
● Windows' 2000
● DOS

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.

5.1 Technical Basics

5.1.1 Block Diagram of the eGrabber-4plus
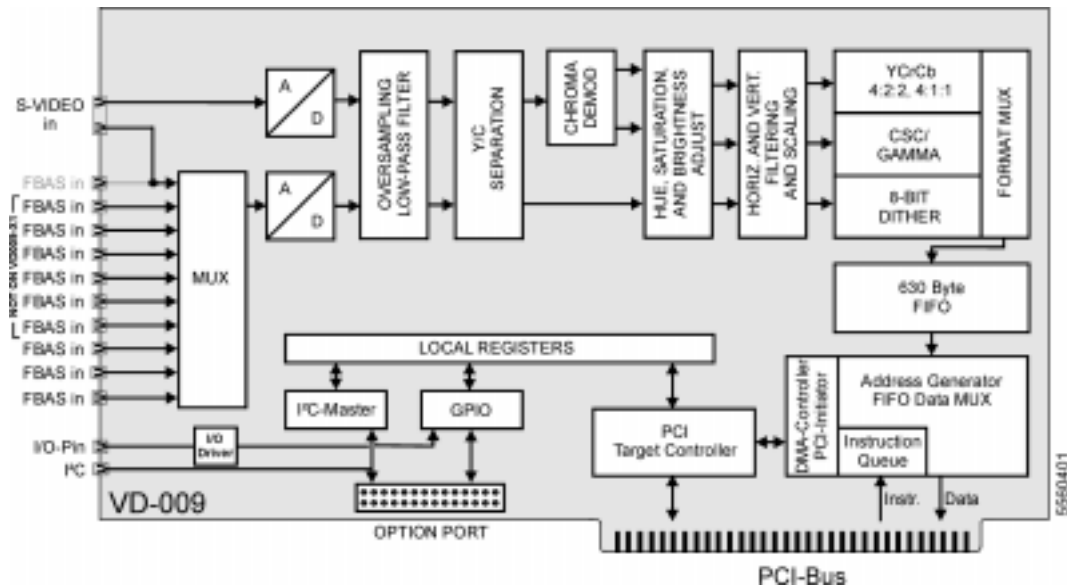
Figure 38:   Block diagram

Figure 38 shows the block diagram of the eGrabber-4plus. 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)".

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

Via the S-VIDEO-input luma- and chroma-signal can be supplied separately (for example from a S-Video-camera or S-VHS-videore-corder). 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 eGrabber-4plus . The processing of gray scale pictures with 256 gray 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 eGrabber-4plus. 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 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 eGrabber-4plus. The principle of the *pixel instruction list* is more closely addressed in *section 5.1.2.*

Via the PCI-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 I²C interface integrated in the Grabber.

5.1.2 The Videosignal and Digitization

The standard videosignal, which is processed by the Grabber, contains 625 lines, which are divided into two fields. 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.

Should a full image be represented, both of the consecutive half images must be interlaced according to *Figure 39* (television via electrical circuitry, in PC-Graphic Memory via software).

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.
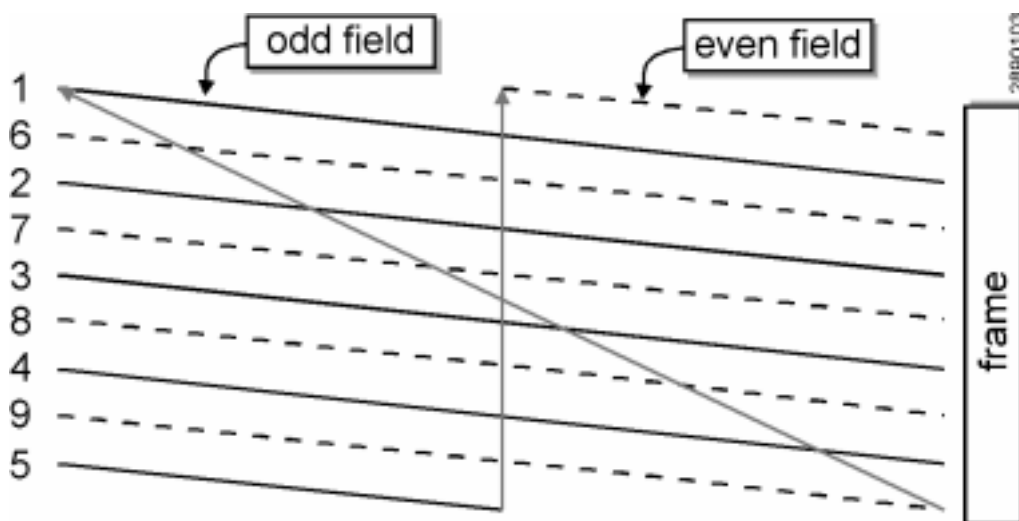
Figure 39:   Interlaced Image (Example with 9 Lines)

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.
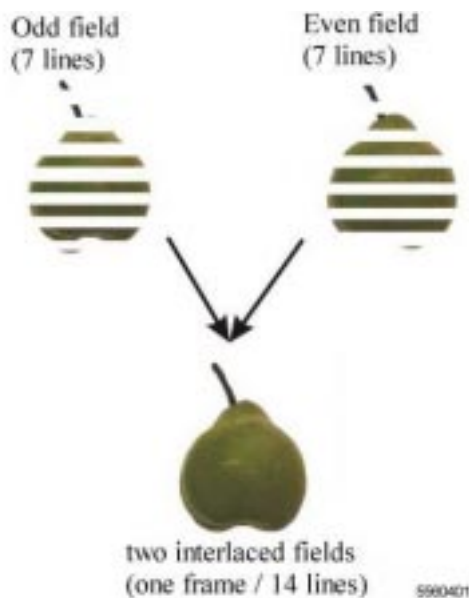
Figure 40:   Fields and Frames

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

5.1.3 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 eGrabber-4plus 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 asYCrCb4:2:2.

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

Caution:
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.

5.1.3.1 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 eGrabber-4plus 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.

Image data is transferred to the Bus's programming memory in real time. This is enabled by the high data transfer rate of the PCI-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 eGrabber-4plus 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 eGrabber-4plus 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 eGrabber-4plus, 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 eGrabber-4plus 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 eGrabber-4plus 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.
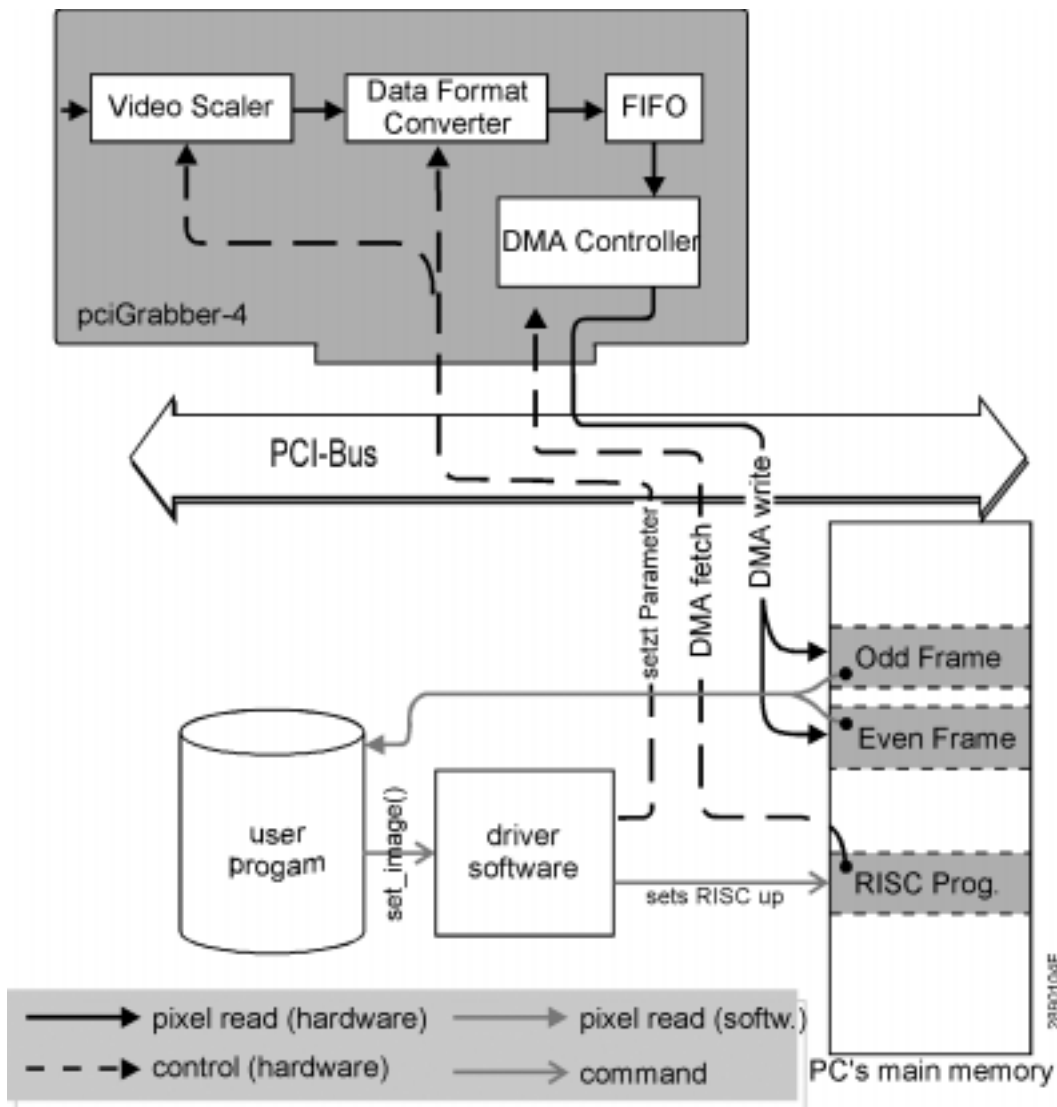
Figure 42: Pixel- and Control Data Flow (Overview)

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 f*igure 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 eGrabber-4plus. 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. Initially this is the wait time until the beginning of the desired half image.

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.

In this case the wait time is almost 40 ms (two half images, worst case).

The image is subsequently digitalized and is complete 20 ms later. For the next 20 ms nothing occurs in this memory space, since during this time the odd-image is delivered.

If continuous digitalization is desired, then image information is written to the memory space of the even image during the subsequent 20 ms.

It is important to note here that in this time frame the old image information in this memory space will be continually replaced by new information. This can lead to false interpretations of the image contents by the software, especially in scenes that have been moved.

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.

5.2 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.
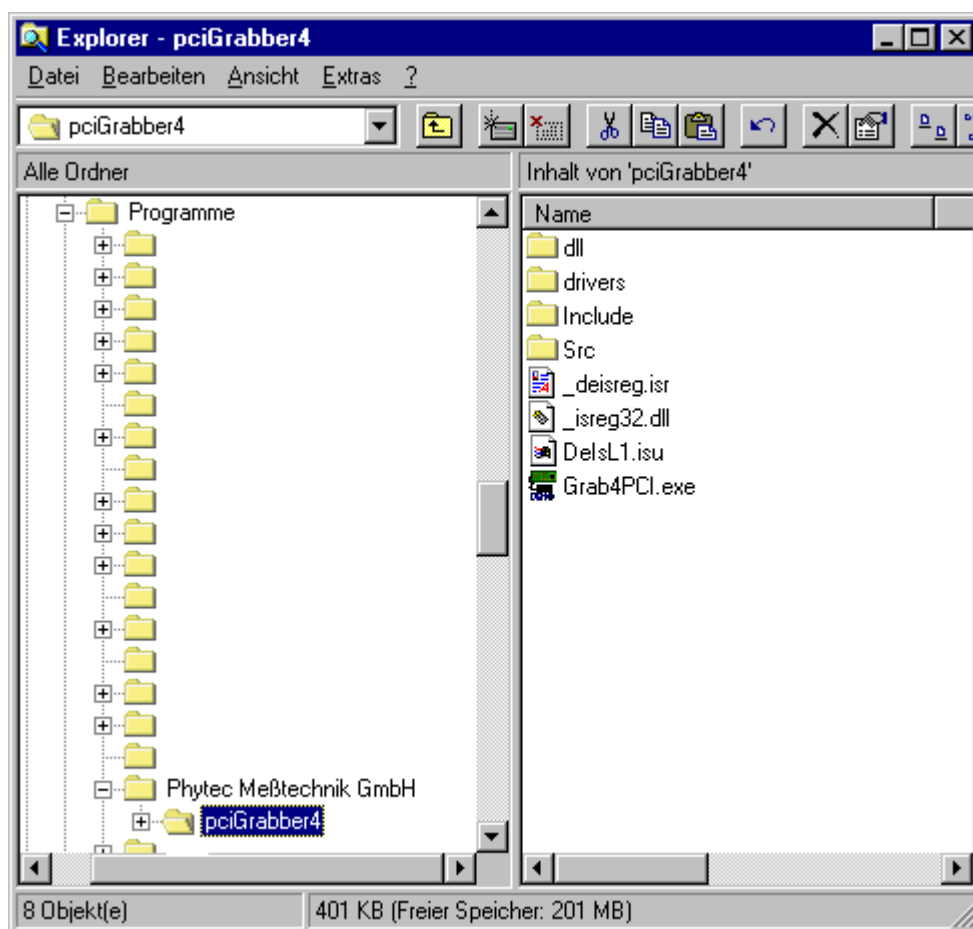
Figure 43:   Directory for Window Driver's New Image.

5.2.1 Requirements

Programs for the eGrabber-4plus can be created with the help of various development environments.

These newly created applications will work with the Windows95/98/ME/XP^{TM} 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 eGrabber-4plus 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.

5.2.1.1 Installation of the VxD-Driver (Windows '95)

The VxD-driver has the purpose to allocate a continuous memory region, which will be the memory for the image. Since the eGrabber-4plus stores the data per DMA-access, it is required, that this physical fixed area has continuous addresses in a sequence.

The reservation of this memory region can only be achieved for Windows'95 with VxD.
By the VxD, the linear memory addresses are converted to physical memory addresses.
The VxD-driver is not called directly by the user program. The access will be executed via the DLL.
You might use the installation CD to deliver the Window's 95 driver with your own application.
The path on the CD is :

PCIGRAB4/DRIVER/WIN95_98.

You can copy these files onto a disk and distribute them with your application. Alternatively you can create your own installation routine. To do this, the following points must be processed:

The VxD-driver must be introduced to the Windows'95-system as s*tatic device-driver* . This will be established in the following way:

- First the file ***PCIGRAB4.VXD*** must be copied to the Windows-system directory.
- In the next step the driver will be included in the *registry-table* :
  Please use the program **REGEDIT** in the **WINDOWS95** - directory. Please step through the registry key tree until you reach *VxD* (*see Figure 44*).
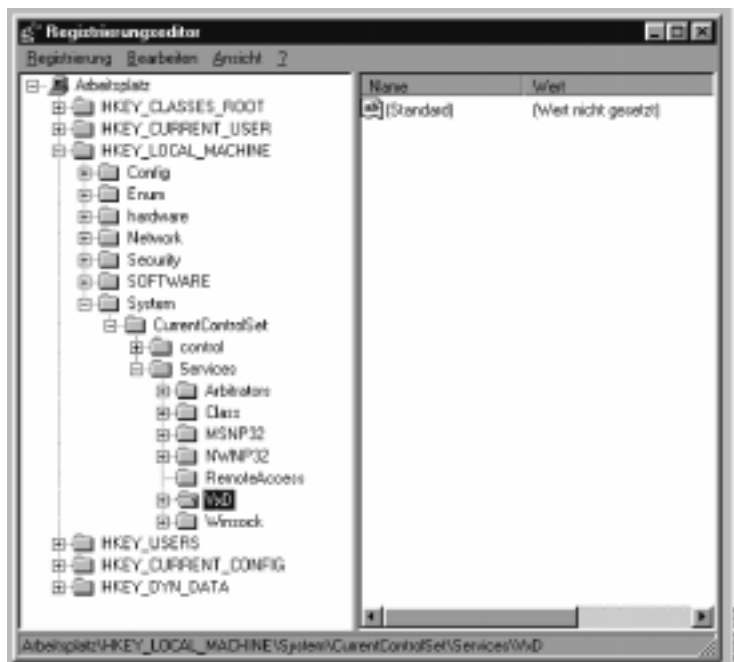
Figure 44:   Windows '95 Registry New Image

Extend the key group *VxD* with „Grab4PCI", by selecting the menu "*Edit - New - Key*'', create a new key and assign the name "Grab4PCI" as shown in *Figure 45*.
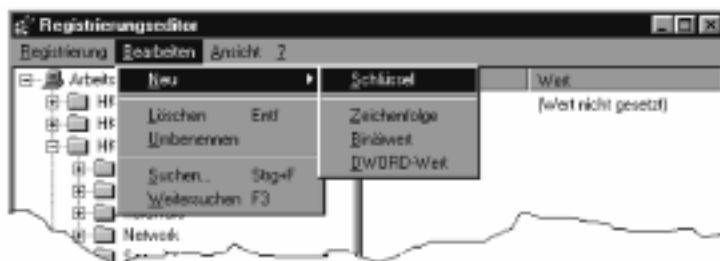
Figure 45:   Adding a VxD-Entry New Image

Now you configure the new Key-group: mark the listing „Grab4PCI" as shown in *Figure 46*. Select from the menu „*Edit - New*" the option „*String Value*". Within the key of „Grab4PCI" a listing will be generated with the notation „*New Value #1*" . Please change this to „*StaticVxD*". Click with the right mouse button the listing and select „*Modify*". Write in the following dialog in the field „*Value*" the character string „*pciGrab4.VXD*" .

Subsequently you select ''*Edit - New*" with the option ''*Binary. Value*" and create as before the binary listing „Start" with the value „00". The final result must look like *Figure 46.*
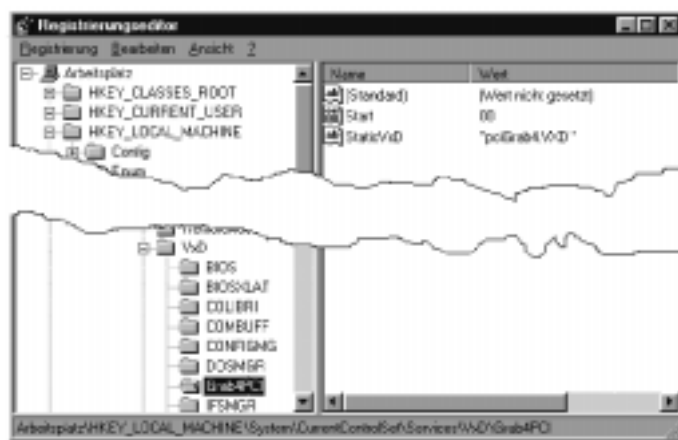
Figure 46:   Configuration of the VxD New Image

● Since Windows'95 accepts the values of the registry-table only after a reboot, you must start the system again.

Caution:
During de-installation of the user program the VxD-driver should be removed. The driver has to be erased from the registry-table and cleared from the system-directory.
The VxD-driver requires for the eGrabber-4plus a region of 1.2 MB in the main memory, which is not available for other applications.
Please be very carefully to change the registry values, since a faulty alteration might destroy the whole configuration. Even your Windows'95 system might not be operating anymore!
To the user, installation and de-installation programs should be provided, so that this process is executed automatically.

5.2.2 Application of the Device Driver for Windows NT4.0

The device driver functions in the same manner as in Windows 95; the driver allocates physical memory to store images. The driver also allows access to the eGrabber-4plus' register.

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 and select *Services* (*see Figure 47*).
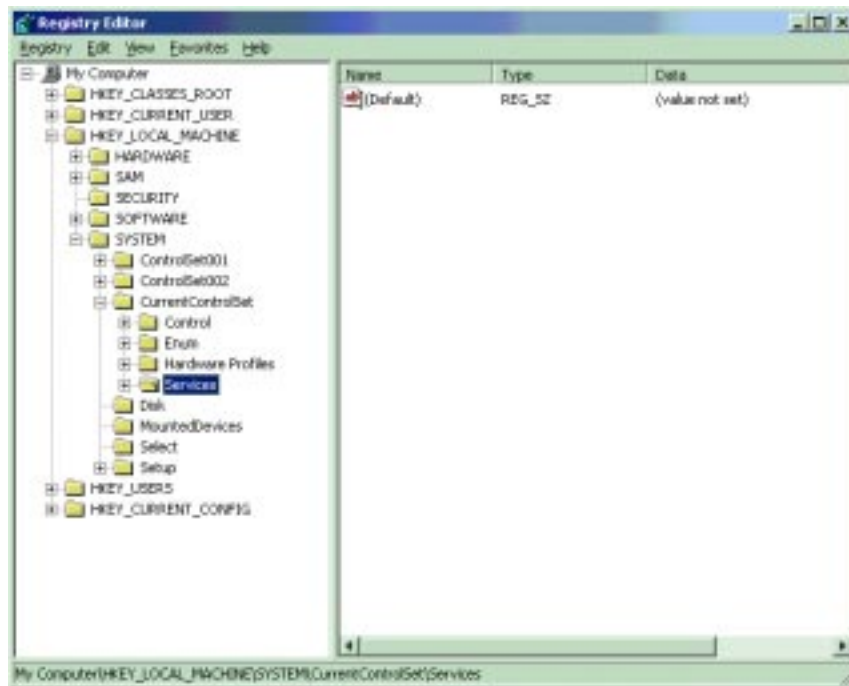
Figure 47: Windows NT Registration Editor

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 48.*
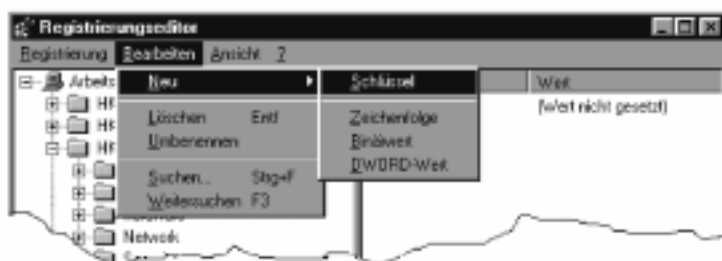
Figure 48: Entering a Device Driver New Image

Now configure the new key group and mark the entry with „*pciGrabber4"* as shown in *Figure 48.*

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 49*.
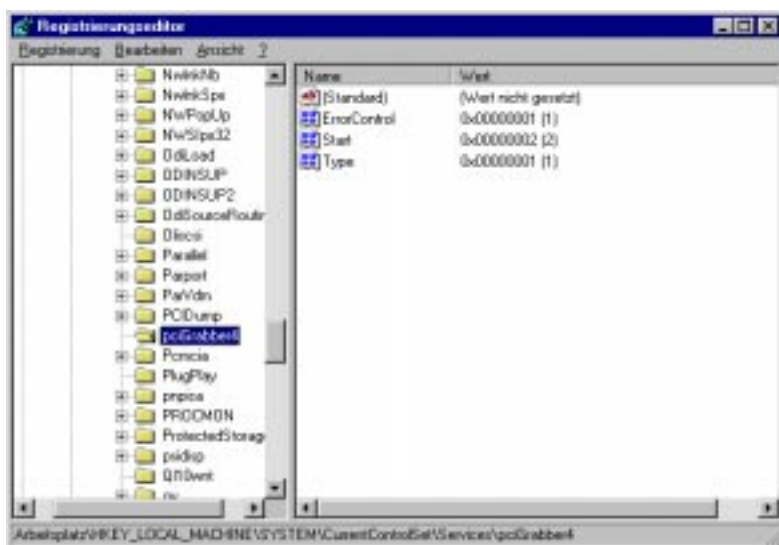
Figure 49:   Configuring the Driver New Image

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 eGrabber-4plus 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.

5.2.3 Application of the Device Driver for Windows' 98 ™ and Windows' 2000/XP

Similar to Windows' 95, the device driver allocates physical memory for storing images. The driver also allows access to the eGrabber-4plus' register.

Reserving memory space is only possible with a device driver under the Windows98/2000 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 Windows98/2000/XP driver with user specific applications.

These applications are located on the installation CD under the **PCIGRAB4\DRIVER\WIN2k_98** directory. Files from this directory can be copied to a disk and applied to user applications.

5.2.4 Application of the DLL

The DLL provides communication between user programs and the eGrabber-4plus. 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.*

5.2.5 Application of the Windows 95/98^{TM} / Windows NT4.0^{TM} / Windows 2000^{TM} DLL

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.

5.2.6 Programming under Delphi

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-
rameter 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';
```

    L-608e_4

```
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.
```

5.2.7 Description of the DLL in Existing Functions

The user can control all of the events in the eGrabber-4plus 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.

❺ 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, I²C interface, etc. These functions need only to be called when a corresponding Grabber feature is implemented.

Most functions are identical in the Windows and in the DOS driver versions. Although, depending on driver model, differences will occur. In order to simplify the porting process, the following symbols are used:

⇔   Calling functions are the same under DOS and Windows.
      Please ensure that the operating system's variable types are distinguishable.

▦   This function is specific to Windows

DOS This function is specific to DOS

Caution:
In all the following routines the parameter `nDevNo` is used. This parameter identifies the desired eGrabber-4plus, in case of several eGrabber-4plus are in the system. The number of the installed eGrabber-4plus can be determined by the function *Max_Device_Number().*

Compatibility to the eGrabber-2

The driver is downwards compatible with the eGrabber-2 (EPC-031 or EPC-031-X1). Programs that have been written for the eGrabber-2 also function with the eGrabber-4plus.

In order to ensure operation of functions for the eGrabber-4plus, new or altered functions have been created for the driver.

Functions that are not compatible with the older driver version for the eGrabber-2 are denoted with a star (☆).

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

Most functions are compatible with the eGrabber-2, 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.

Compatibility with the eGrabber-2*plus*

The eGrabber-4*plus* is fully compatible with the eGrabber-2*plus* (VD-009 / VD-009-X1). Applications that were written for the eGrabber-2*plus* will run without limitations on the eGrabber-4*plus* as well.

⇔ ● 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 diver 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.
This function is only for the Windows-DLL and not for DOS.

Note:
Test the version number and ensure that when using the eGrabber-4plus the Major_Version_Number is larger than or equal to 5.

⇔ ❶ Determination of the number of available eGrabber-4plus

WORD Max_Device_Number (void);

Returnvalue: Number of eGrabber-4plus found

This functions specifies the number of eGrabber-4plus in the system. This is required, because PCI-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 1.4).*
The user has not to care about addresses or address regions when using the driver. Those are converted internally into *device numbers* (= nDevNo).

Each eGrabber-4plus 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 eGrabber-4plus 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 eGrabber-4plus 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 eGrabber-4plus.

Caution:
For nDevNo only values between

$$0 < nDevNo <= \text{Max\_Device\_Number}()$$

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 the Grabber. 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 Calling-Up 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.
`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) = 0x03`. Therefor, Grabber number 1 is EPC-32.

MAX_CHANNEL :   Returns the number of composite input channels.
For example, for EPC-32 = 3, for EPC-32-X1 = 3.

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 eGrabber-2 (EPC-032), eGrabber-2 (EPC-032) or a related product is called up, then the error code 6 and the string „EPC-032 or compatible" is returned.

⇔ ❷ Grabber setting to the required 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_BDGHI` configures the Grabber for the application of PAL-videosources.
❍ `NTSC_M` configures the Grabber for NTSC-sources

⇔ ❷ 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.

⇔ ❷ 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-023 cable connected to sockets ❶ and ❷. 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*.

⇔ ❷ ☆ Configuring the S-Video Mode

BYTE Set_S_VideoEx(WORD nDevNo, BYTE input);

Input: MINIDIN   = Mini DIN socket is the input
            COMBI   = Combi plug (socket X3) is the input
            AUTO   = automatic configuration

Return
            (a) input = MINIDIN or COMBI
            SUCCESSFUL   = no error occurred
            NOT_SUPPORTED   = eGrabber-2, EPC-031 and
                                                the COMBI parameter
            (b) input = AUTO
            MINIDIN   = Memory storage from the Mini-DIN
                                                socket
            COMBI   = Memory storage of the COMBI socket
            NO_SIGNAL   = signal not found (AUTO mode)

The video processor's second ADC must be activated when connecting an S-Video source. The function **Set_S_Video** activates the *chroma ADC*. The excessive luma notch filter in the luma path is deactivated, creating a sharper image. **Set_S_Video** also automatically connects the input channel to the S-Video input.
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 the cable.

Caution:
The S-Video source can either be connected to the Mini-DIN socket or alternatively to socket X3. **The S-Video source can not be connected to both sockets simultaneously!**

The function has a hand-over parameter that specifies which socket the S-Video camera is connected to. If AUTO is given as a parameter, then the driver searches for the S-Video camera. The driver first tests the Mini DIN socket for an active signal.

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 (socket X3). 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:
- If there is no connection of an S-Video source, but there is a composite source available at channel 3 (EPC-032), or channel 9 (EPC-032-X1), then the Grabber is configured to the Combi socket, using the AUTO function. Then the image from the composite source is displayed in black and white.
- 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.

⇔ ❷ ☆ Configuring the Input Channels

void Set_ChannelEx(WORD nDevNo, WORD nChannel);

nChannel: Input channel to be configured (1…9 / 1…3)

This function is used to select the input channel. The signal is directed by a dual-row multiplexer. The first row is externally located on the Grabber board, and the second row is integrated into the video multiplexer.
Values:

EPC-032 :   Channel numbers 1 through 3 are allowed
EPC-032-X1:   Channel numbers 1 through 9 are allowed
EPC-032-X2 :   Channel numbers 1 through 3 are allowed
EPC-032-X1-X2: Channel numbers 1 through 9 are allowed
This function is not compatible with the eGrabber-2 (EPC-032) and eGrabber-2 (EPC-031).

Note:
Configuring the input channels is not necessary when using S-Video sources! The function *Set_S_VideoEx()* switches the channels automatically.

Caution:
When switching input channels, stop inhibit times are to be taken care of until an image for the new channel has been digitized,.

For the following reasons,

- Definition of the video signal normally lasts up to 4 half images 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.

The function also supports the older models. When using the older models, the following hand-over parameters are allowed:

EPC-032 and EPC-031:   Channel numbers 1- 9 allowed
EPC-032-X1and EPC-031-X1:   Channel numbers allowed:

1 = input 1
5 = input 2
9 = input 3

⇔ ❷ **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.

⇔ ❷ 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 artifacts from the motion.

When displaying only fields, a 20 ms recess usually occurs between the two digitization events. This is because the other field cannot be displayed due to the field rastering delay. In the *Field Aligned* mode, the second field 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 field is altered during the electronic filtering, this function is only suitable in a range limited for measuring and automation tasks.

⇔ ❷ De/Activation of the AGC

**void Set_AGC(WORD nDevNo,WORD nCAGC, WORD nAGC,
WORD nChrush);**

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 eGrabber-4plus 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 modus *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 eGrabber-4plus with a non-adaptive power regulator will suffice.

In applications where the regulator can have damaging effects on the **adaptive** AGC (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 AGC should never be switched off, because it acts like a clamping device.

⇔ ❷ 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 gray 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

The eGrabber-4plus 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 one image is omitted. Images 26-49 are produced and then again one image is omitted.

● *nDecField=1, nDecRate=25*
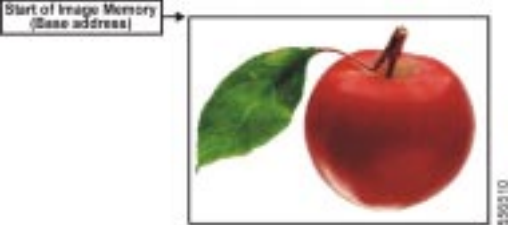
In this case 25 fields are omitted of 50. The result will be, that each second image will be produced, so always the same field type will be omitted. Which field will be omitted first, depends on nAlign.

▦☆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 fuction, the vertical orientation of the image in the memory can be set:

flip = 0	flip=1 (default)
The image is stored upright. This means, the lowest image memory address contains the <u>upper</u>-left corner of the image:	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:
	  This might be useful, if image date are processed in BMP-format.

Important:
- The setting of the vertical image orientation with *FlipPicture()* must be done before the *Set_Image()* – function ic called. The settings takes affect 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.)

❷ 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.

Be aware that this function has different parameters in the DOS version of the driver (see below).

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* ⇔ ❷ *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 50* and *Figure 51*demonstrate the significance of the parameter.

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

If the section defined by *hsize* and *vsize* 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.
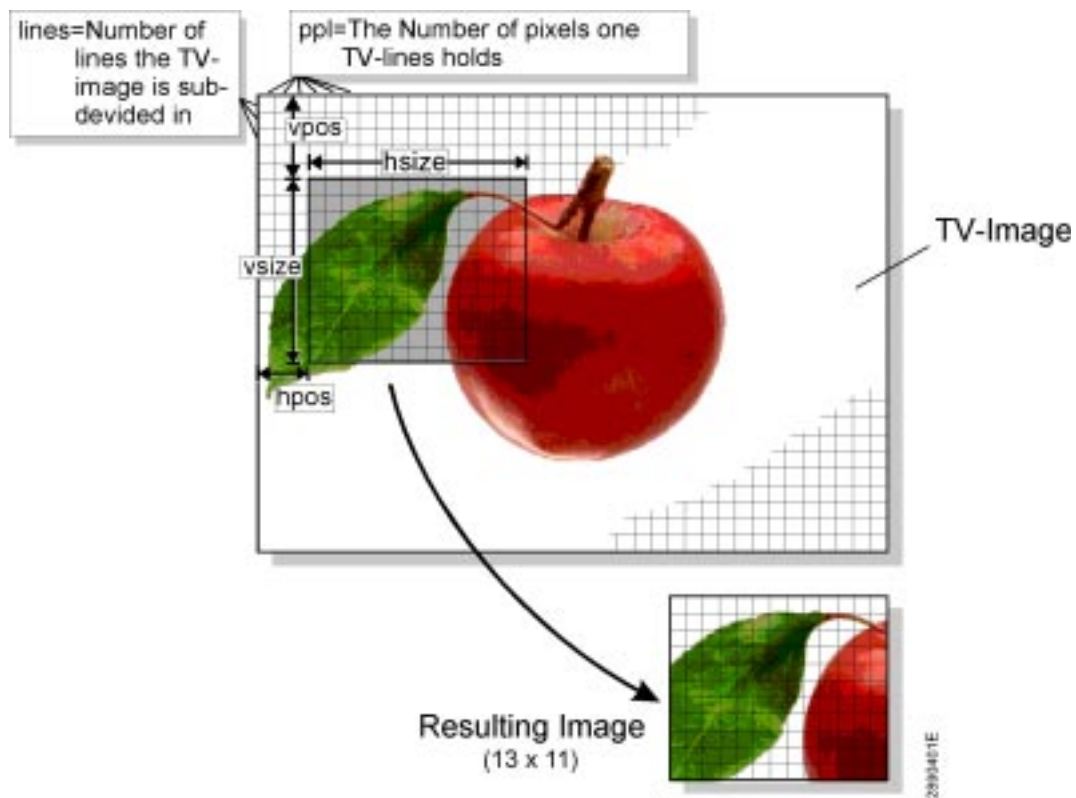
Figure 50:   Scaling and Cropping

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.
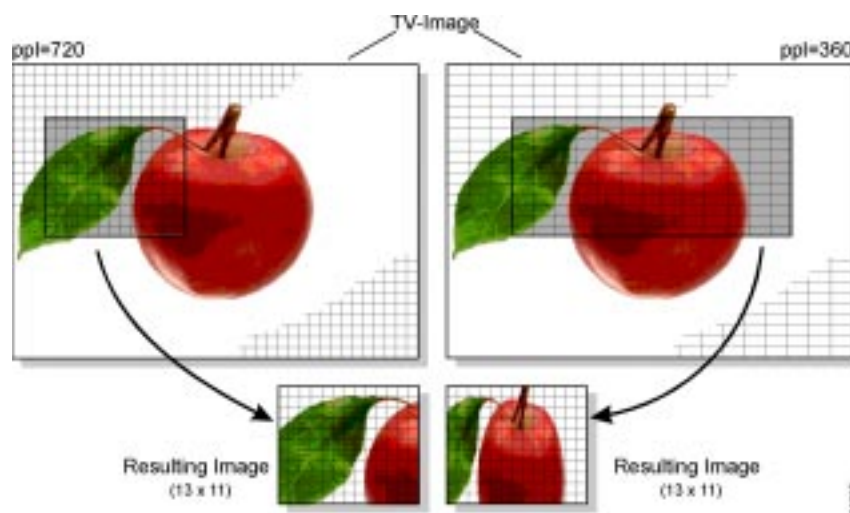
Figure 51:   Example of Scaling: Only the ppl Value is Different

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 = ½ lines
  nOlines = ½ 576 = 288
  nElines = ½ 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 = ½ 500 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 (even value)

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

nOvpos = nEvpos = 18 (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 52* 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 52 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 ($2^{38} = 16.777 \bullet 10^6$).
  - *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 52* 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 52*).

● **YCrCb:** In this format we find gray 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 52*. 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).
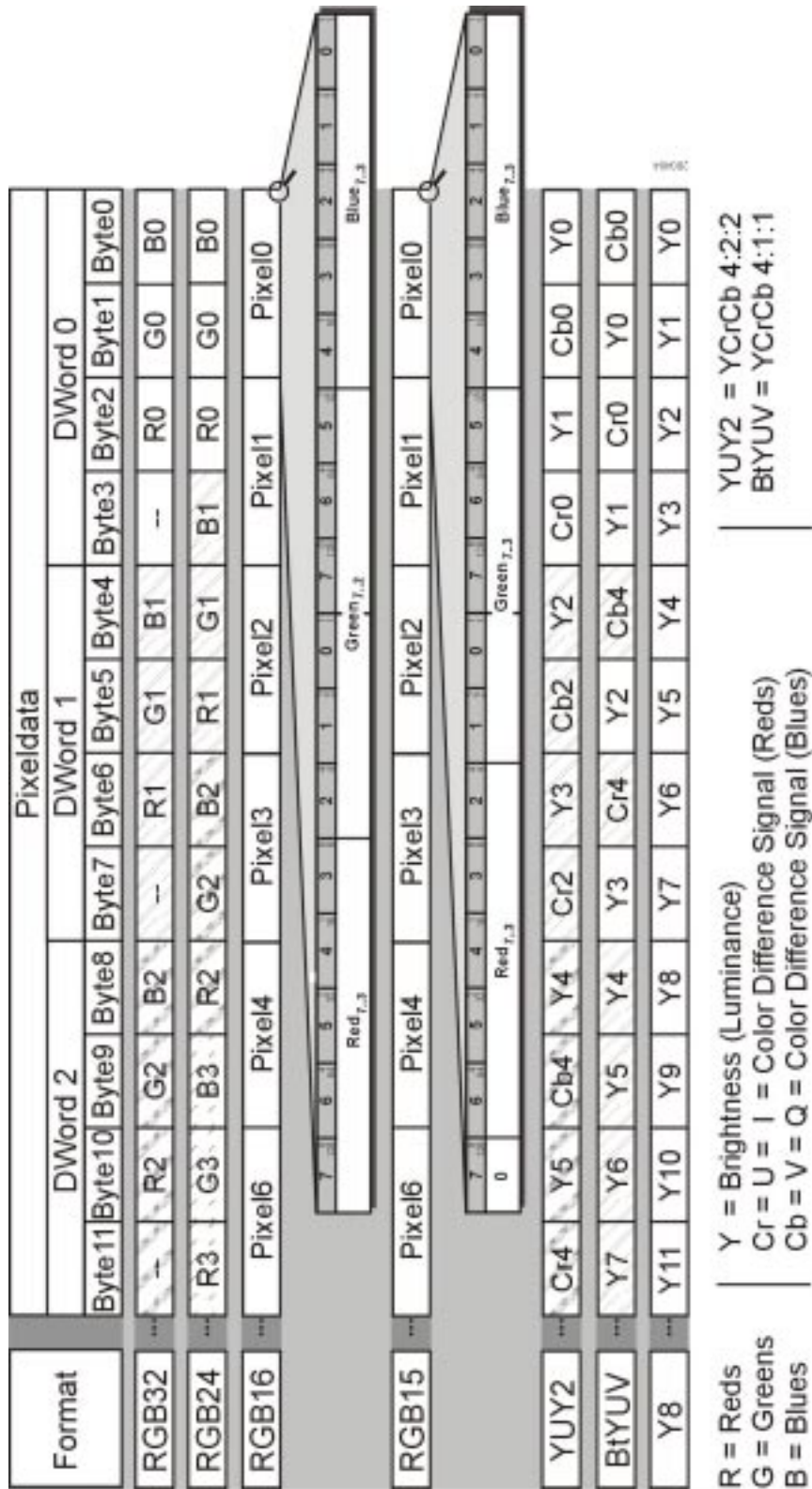
Figure 52:   Color Format of the eGrabber-4plus

- **Y** : In the gray scale format only the gray 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 gray 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.

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 · vsize · pixel size [Byte]

The value pixel size can be depicted from *Table 12*.

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	*pixel size* [Byte]
RGB32	4
RGB24	3
RGB16, RGB15	2
YUY2	4 Byte per 2 pixel
BtYUV	12 Byte for 8 pixel
Y8	1

Table 12:   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) alternatingly. 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 a processed alternatingly.

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*).

⇔ ❸ 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.

⇔ ❸ 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 (see *Figure 49*).

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** .

⇔ ❸ **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 53*).
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.
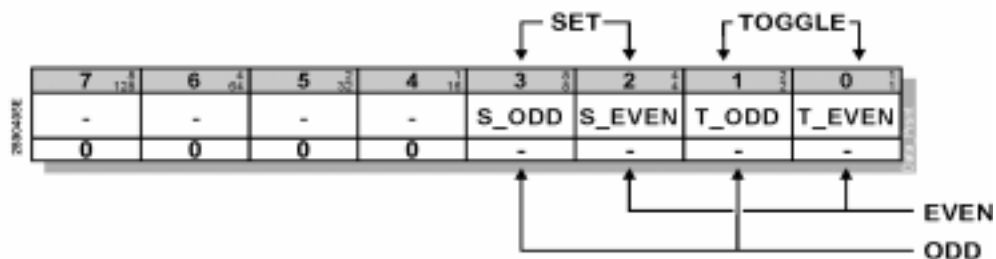
Figure 53:   Return Values of ‚Data_Present'

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.
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-bus, which might hinder the data transfer of the Grabber. You might include delay times between inquiries, in order to avoid slowing down digitization.

Caution:
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.
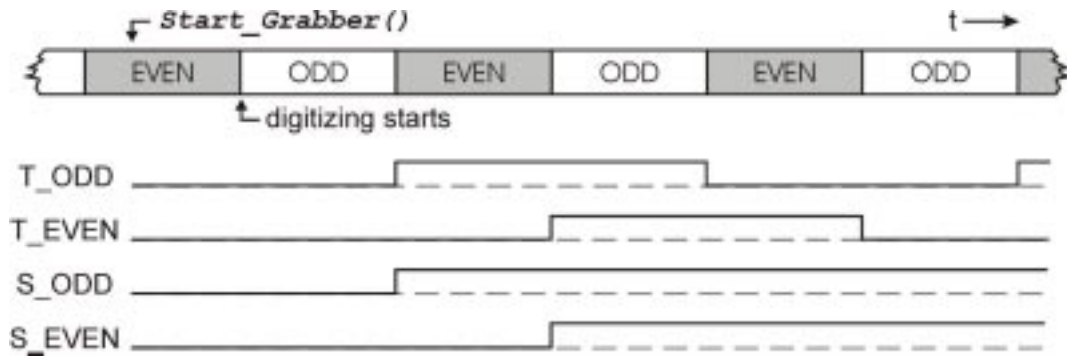
*Figure 54:   Timing Diagram of the Return Parameter of ‚**data_present**()*

▦ ❸ 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().**

Caution:
When using the **Windows'95** driver, all Grabbers in the system use the same image buffer. For this reason, two Grabbers cannot simultaneously digitize images when the standard routine for the driver is being used. If both Grabbers are in the digitization process, then image data stored in the buffer will be over written.
This limitation does not exist for drivers under all the other operating systems.

⊞ ❸ ☆ 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 ist 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 Windowa 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 happend 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 advisabe 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 continuos digitization this is done by evaluatiing 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 oft 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 applicatin by the operation system might last a noticable (and variable) time.

⇔ ❸ Checking for a video signal

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 2, 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 ±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.

⇔ ❸ ✫ **Number of Processed Digitized Images**

With the eGrabber-4plus, 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 eGrabber-2.

BYTE Get_CaptureCounter (WORD nDevNo)

return value: Number of grabbed fields module 256

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

void Reset_CaptureCounter (WORD nDevNo)

Calling this routine sets the half frame counter back to zero.

⇔ ❹ **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 =   brightness[%] ⋅ 2.5601 [1/%]

⇔ ❹ **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.

⇔ ❹ **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 =   contrast [%] · 2,1598 [1/%]

⇔ ❹ **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.

⇔ ❹ **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 = U-saturation [%] · 2.5400 [1/%] ; 0% ... 201.18 %
nSat_V = V-saturation [%] · 2.1396 [1/%] ; 0% ... 238.83 %

⇔ ❹ **Reading the content of the registers for 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.

⇔ ❹ Correction of the hue (only for NTSC)

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°, therefor the color signal can be varied in the range of -89.3° to +90°
This value must be set to 0 to ensure proper functioning of the color decoder under PAL.

⇔ ❹ Reading the content of the registers for hue

short Get_Hue(WORD nDevNo);

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

This function provides the hue value.

⇔ ❹ 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 (= ½ whole frame), QCIF ($^1/_4$ whole frame) and ICON ($^1/_8$ whole frame) .

Default: luma-low-pass is turned off.

⇔ ❹ 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.

⇔ ❹ 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 gray 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.

⇔ ❺ **Reading/Writing data via the option port (GPIO-Port)**

12 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: Direction control for GPIO-Port (0…4095), bit encoded
nData: Data that should be sent out over the GPIO-Port

return value Get_GPIO_Data(): Data that is read by the GPIO-Port

The eGrabber-4plus 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 12-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.

⇔ ❺ ☆ Transmitting Data via the I²C Interface

With these functions it is possible to read and write to components that are connected to the I²C interface.
Instructions on how to connect components or devices to the I²C-Bus can be found in s*ection 1.7.*

Caution:
The I²C-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   baud rate
         eGrabber-4plus: 0 = 99,2 kHz, 1 = 396,8 kHz
         eGrabber-2:   0 = 33 kHz, 1 = 290 kHz

This function determines the baud rate for transmission on the I²C 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 I²C device on the bus
bSubAddress: Internal memory address for the I²C 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 I²C 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 I²C 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 I²C device on the bus
bSubAddress: Internal memory address of the I²C 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 I²C device.
The function returns an error code as a return value.

Note:
The internal processing time of some devices (EEPROMs for example) might be noticable longer than the I²C-bus cycle time. If several write cycles are sent to such a device, this might result in a NOACK-error, since the device is not yet ready to process the next write cycle.
To avoid this, read commands should be executed after a write cycle, until the written value is returned. After that, the device is ready to process a new write cylce. *Please refer also to the data sheet of the I²C-device used.*

⇔ ❺ ☆ **Using Internal EEPROM**

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

Note:
Since the predecessor models to the eGrabber-4plus 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 bSubAddres, 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.

⇔ ❺ ☆ **Controlling the I/O Pins**

The eGrabber-4plus offers an externally available, transistor driven I/O pin. Using control signals, this I/O pin is freely selectable as input or output. Both of the following functions can be used to control the I/O pin.
Instructions on how to attach devices to the I/)-Pin can be found in *section 1.6*.

This function is not available for the eGrabber-2.

void Set_Ext_IO (WORD nDevNo, BYTE bData)

bData: Control value, 0 = CLOSE, 1 = HI-Z

This function controls the switch output of the port pin. The output pin's transistor operates as a switch against Ground. If the control value is set to 0, the switch is closed and current is supplied to the I/O pin.

If the control value is set to 1, then the transistor is disabled (high ohm / switch opened).

For more information on the technical specifications for the I/O output, please *refer to section 1.2.*

Note:
When starting the computer, the transistor is disabled (HI-Z). Connecting an external pull-up resistor creates a logic „1" signal. A connected device, i.e. a relay, has no power (switched off).

BYTE Read_Ext_IO (WORD nDevNo)

nDevNo   = Device Number

return value: State of I/O pin

This function allows the state of the I/O pin to be read. This enables the pin to be used as input.
If the voltage connected to the I/O pin is in the range of „logic 0", then the function returns a value of 0. If the voltage is in the range of „logic 1", then the function returns a value of 1. For the appropriate voltage ranges, please *refer to the specificati ns in section 1.2.*

Caution:
In order to use the I/O pin as input, the transistor must be disabled (HI-Z).

⇔ ❺ **Direct reading and writing from/to 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: contents of the register

Almost all functions of the eGrabber-4plus can be controlled by routines of the driver. We recommend explicitly the utilization of the standard functions.

In case the user wants to access the registers of the eGrabber-4plus directly, the following functions are available with which you can write to and read from each register of the video processor.

If you need more information on the registers of the eGrabber-4plus please contact the PHYTEC-Support.

Caution:

For the existing configuration, some registers of the eGrabber-4plus must be configured with special values, in order to guarantee the proper operation of the board.

Please check before each alteration the exact signification of the changed register / bits and the effect.

PHYTEC is not responsible for any damage, which might result due to manipulations of the registers.

5.3 Driver for DOS Applications

The PCI4GRAB driver library, can be found on the CD's directory, under **PCIGRAB4\DRIVER\DOS\DRIVER**. The eGrabber-4plus can be adapted to DOS using this library.

5.3.1 Premises

Under the DOS operating system, it is required, that for the operation of the eGrabber-4plus, the whole physical address area of the PC can be addressed linearly. This is necessary, because the PCI-BIOS configures the register area of the Grabber in the memory just according to its own rules and the user has no influence. Usually high addresses are utilized above the RAM region, which require special addressing mechanisms.

Caution:
In order to utilize the DOS-driver, the DOS-Extender *DOS/4GW* has to be installed. Programs, which drive the eGrabber-4plus under DOS via the driver routines, have to be designed in such a way, that they can operate with *DOS/4GW*.

DOS/4GW from Rational Systems is a DOS-Extender, which provides protected-mode-access. Many compilers support the application of DOS/4GW and allow combining it with EXE-files of the user program.

Caution:
Some DOS-system programs may cause problems working together with DOS/4GW. Especially the utilization of **EMM386.EXE** might be problematical. Therefore we recommend not to use DOS/4GW and **EMM386.EXE** simultaneously.

5.3.2 Development Platform

The DOS driver software is object-oriented according to C++. It is possible to link the driver also to programs, which are written in a different programming language. In this case you have to consider the corresponding procedures to call these programs.

The condition for using this library is, that the compiler supports 32-bit code applications, since this library of the eGrabber-4plus is a 32-bit library.

For the memory model you have to select *32-bit-Flat*. We recommend the setting of the option '*80386 Register Based Calling*'.

For the programming of the driver, the compiler Watcom C/C++ from Powersoft version 10.6 was used. This compiler allows the 32-bit programming and the linking of DOS/4GW.

The driver is delivered as library (***.LIB**) - file and can be linked with all compilers, which support the used format. The header files (***.H**) are included in the user program. They provide all declarations of the different functions, which are provided from the library.

During the start of the user program the DOS/4GW-extender must be loaded, if the compiler does not automatically link the extender and therefore triggers the automatic start. DOS/4GW allows the DPMI-access, which is required for the functioning of the driver routines.

5.3.3 Functions for the DOS Driver `PCI4GRAB`

The PCI_GRABBER4 class contains all of the functions for initialization and configuration, and can be found in the library **PCI4GRAB.LIB**. These functions also control the Grabber events.
With an object from this class, all Grabbers in the system are controlled.
For easier identification, all of the functions have been divided into five groups. The number of each group is shown in a black circle.

The functions are classified as follows:

❶ Routines for Initialization
This group of functions must be called before using the Grabber to ensure that the Grabber operates correctly.

❷ Routines the Grabber configures for the grabbing process:
Functions from this group configure the Grabber to the connected image source (camera). These functions also determine how the grabbed picture will appear 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 are also called-up several times during 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.

❺ Routines for Controlling the Option Port
This category includes functions that do no directly influence the grabbing process, but rather deal with the features of the Grabber, i.e. I/O port, I²C interface, etc. These functions need only be called when a corresponding Grabber feature is implemented.

Most of the functions are identical for the Windows and DOS driver versions. Although, depending on driver type, a few differences do exist. In order to simplify porting from programs, the following symbols are used:

⇔ Calling the function is identical for DOS and Windows
Please ensure that the operating system's variable type can be distinguished.

⌨ This function is Windows specific

DOS This function is DOS specific

Caution:
In all of the following descriptions for routines, the parameter `nDevNo` is used. This parameter identifies the desired eGrabber-4plus when multiple Grabbers are installed in the system. The number of installed Grabbers can be determined by the function **Max_Device_Number**().

Compatibility to the eGrabber-2

The driver is downwards compatible with the eGrabber-2 (EPC-032 and EPC-032-X1). Programs that have been written for the eGrabber-2 also function with the eGrabber-4plus.
In order to ensure operation of functions for the eGrabber-4plus, new or altered functions have been created for the driver.
New programs that use these altered functions cannot operate with the eGrabber-2.
Functions that are not compatible with the older driver version for the eGrabber-2 are denoted with a star (☆).
If the new features of the eGrabber-4plus are to be added to existing applications, please take note of the features denoted with a star (☆).

❺ **Get Error Message**

short Get_Error(void);

return value: 0 = no error
                  1 = device number not found
                  2 = bad register number
                  3 = initialization failed
                  4 = Grabber not found
                  5 = unknown parameter value

⇔ ➊ **Determination of the number of available eGrabber-4plus**

unsigned short Max_Device_Number();

return value: number of the eGrabber-4plus found.

⇔ ➊ **Initialization of the Grabber and driver activation after power up**

void Initialize(unsigned short nDevNo);

⇔ ❶ Get Information about the Grabber Cards installed

short Read_GrabberInfo(unsigned short nDevNo,
unsigned short wInfoType)

wInfoType: specifies the type of information requested

return value: information requested

⇔ ❶ Get Grabber name as text string

short Read_OrderCode(unsigned short nDevNo,
unsigned char* sCodeString,
unsigned long dwSizeOfString)

*sCodeString: pointer to a string variable (must be defined previously) (25 ytes min.). The function will write the grabber name into this string variable.
dwSizeOfString: size of the string array

return value: error code

⇔ ❷ Grabber setting to the required color-system

void Set_Color_System(unsigned short nDevNo
unsigned short nColSys);

nColSys: code for TV color system

⇔ ❷ Recognition of the video format

short Get_Video_Status(unsigned short nDevNo);

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

⇔ ❷ Configuring the Composite Mode (Composite Inputs)

void Set_Composite(unsigned short nDevNo);

⇔ ❷ ✩ Configuring the S-Video Mode

unsigned char Set_S_VideoEx(unsigned short nDevNo,
                              unsigned char input);

input:	MINIDIN	= Input is a Mini DIN socket
	COMBI	= Input is a Combi plug (HD-DB-15 plug)
	AUTO	= automatic configuration

return value:	MINIDIN	= Stored from Mini DIN socket
	COMBI	= Stored from Combi socket
	NO_SIGNAL	= signal not found (AUTO mode)
	NOT_SUPPORTED	= eGrabber-2 and COMBI configuration

⇔ ❷ ✩ Configuring the Input Channel

void Set_ChannelEx(unsigned short nDevNo,
                     unsigned short nChannel);

nChannel: Input channel to be configured (1..9 / 1..3)

⇔ ❷ Selection of the luma notch filter for black/white -operation

void Set_BW(unsigned short nDevNo, unsigned short nOn);

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

⇔ ❷ De-/Activation of the Interlaced Mode

void Set_Interlace(unsigned short nDevNo,
 unsigned short nInterlace);

nInterlace: 0 = Non-Interlace
       1 = Interlace

⇔ ❷ De/Activation of the AGC

void Set_AGC(unsigned short nDevNo,
    unsigned short nCAGC,
    unsigned short nAGC,
    unsigned short 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

⇔ ❷ De/Activation of the color killer

void Set_CKill(unsigned short nDevNo,
     unsigned short nCKill);

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

⇔ ❷ Ignoring fields/frames in a video signal

void TemporalDect(unsigned short nDevNo,
unsigned short nDecField,
unsigned short nFldAlign,
unsigned short 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

❷ DOS Setting the size and scaling of the image

void Set_Image   (unsigned short nDevNo,
unsigned short nOhpos,
unsigned short nOvpos,
unsigned short nOhsize,
unsigned short nOvsize,
unsigned short nOppl,
unsigned short nOlines,
unsigned short nOColformat,
unsigned char *pOImgBuf,
unsigned short nEhpos,
unsigned short nEvpos,
unsigned short nEhsize,
unsigned short nEvsize,
unsigned short nEppl,
unsigned short nElines,
unsigned short nEColformat,
unsigned char *pEImgBuf,
unsigned short nColSystem,
unsigned short nInterlaced,
unsigned short 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)
pOImgBuf:	address of the odd-image memory
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)
pEImgBuf:	address of the even-image memory
nColSystem:	code for color system (see *Set_Color_System*)
nInterlace:	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 that the picture will be stored later in the memory, will be defined, and the address of this memory region is determined.

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.

Please be aware that the parameters for these functions can be differentiated between the DOS and Window's driver versions.

Under the DOS operating system, the user must reserve a space in the computer's RAM for storing images . A pointer that indicates the beginning of the reserved memory space must also be handed over.

The control of parameters for resolution scaling and positioning is equal to the procedure in the windows.

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(…)*).
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 · vsize · pixel size [Byte]

The value *pixel size* can be depicted from *Table 12*.

For the format YUV2 and BtYUV it must be considered, that 2 or 8 pixels are combined 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)*.

Format	*pixel size* **[Byte]**
RGB32	4
RGB24	3
RGB16, RGB15	2
YUY2	4 Byte per 2 Pixel
BtYUV	12 Byte for 8 Pixel
Y8	1

Table 13: Memory Requirements for a Pixel in the Single Mode

If small formats are often required, we recommend to apply separate fields. In this case you allocate two memory regions with two pointers providing the start addresses. They are transferred to the function by the parameters *pOImgBuf* and *pEImgBuf* for odd- and even-fields. The parameter *nInterlaced* is set to 0, in order to indicate, that the fields are stored in separate memory regions.

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.

In this case the start address of the frame in the memory is transferred by the parameter *pEImgBuf*.

Note:
This region must contain both fields, so that the region must be twice in size of one field. The odd parameter *pOImgBuf* is not used in the interlaced mode (*nInterlaced*=1).

If only one field (even or odd) should be digitized, the pointer of the field, which is not digitized is set to NULL. Therefore if only one even field is required you have to set *pOImgBuf* = NULL.

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 to the other region (another 20 ms) alternatingly.

This means, that each field memory region is at least not accessed from the Grabber for at least 20 ms.

For frame mode (*nInterlaced* = 1) the Grabber stores continuously to the common memory, 20 ms the odd and then 20ms 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, the image data can be grabbed in two separate half image memory fields and analyzed in an alternating fashion (synchronization in a 20 ms cycle for image actualization).

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*).

⇔ ❸ **Start grabbing**

void Start_Grabber(unsigned short nDevNo);

⇔ ❸ **Stop grabbing**

void Stop_Grabber(unsigned short nDevNo);

⇔ ❸ **Show digitization status**

short Data_Present(unsigned short nDevNo);

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

DOS ❸ ☆ **Interrupt Configuration**

WORD Set_Interrupt(WORD nDevNO, DWORD nInterrupt);

NInterrupt: OFF = No interrupt generation (default)
              VSYNC = Interrupt at the end of a half image

Return Value: Error Status

The eGrabber-4*plus* can generate an interrupt, if the end of a half-image is recognized at the video input. If a digitalization procedure is active, then it can be determined, with the aid of the interrupt and without polling (see Data_Present), when this procedure is finished.

A digitalization procedure can only be complete at the end of a half image, which is signaled by the interrupt. However, an interrupt doesn't absolutely mean that a digitalization procedure is complete, because:

- It may be the case that no digitalization procedure was active (the Grabber is in rest state).
- The most recently completed half image is not the one that is to be digitalized (for example even instead of odd)

- During full image digitalization first a half image appeared or the digitalization has not yet begun, since a leading second half image (even) is still expected.
- The interrupt was possibly generated by another device on the PCI-Bus, which is mapped on the same interrupt channel.

Therefore in the event that an interrupt occurs it is always necessary to test, by means of a query to Data_Present, what the cause of the interrupt was and whether this event was the one expected.
.

The interrupt function therefore represents an expansion of the *Data_Present* function, which minimizes unnecessary *Data_Present* queries. However, it doesn't replace the *Data_Present* function.

Note:
Be aware that interrupts are constantly being generated as soon as a video signal is connected to the preset video input. The interrupts occur in 20 ms intervals. Unusually strong interference in the input signal as well as switching input channels can also generate an interrupt.
The interrupt function should only be switched on during an active digitalization procedure, in other words in the space between *Start_Grabber* and *Stop_Grabber*.

The eGrabber-4plus generates the first interrupt INTA in its PCI-socket. Depending on the socket where it is located as well as the arrangement of interrupts resulting form the BIOS, this hardware interrupt is assigned an IRQ-number. This IRQ-number can be queried via the operating system.

Caution:
First arrange an interrupt routine that can handle the IRQ generated by the Grabber. Only then should you generate the interrupt.
In order to reset the interrupt flags in the Grabber after an interrupt has occurred, the function *Reset_Interrupt* must be called up in the interrupt routine.

> **Note:**
> Using interrupts is not recommended in most applications. Often one is able to achieve the same results with polling (repeated queries to *Data_Present*). Using interrupts may be appropriate if saving processing time is an absolute priority.

DOS ❸ ☆ Resetting the Interrupt Flag

WORD Reset_Interrupt(WORD nDevNo);

Return Value: Error Status

After recognition of the interrupt it is necessary to reset the interrupt flag. In order to do this the function Reset_Interrupt has to be called up from the interrupt handling routine.

⇔ ❸ ☆ Number of Digitized Images

The eGrabber-4plus allows the user to count the number of digitized images. The following two functions are available for this purpose. These functions are not compatible with the eGrabber-2

unsigned char Get_CaptureCounter (unsigned short nDevNo)

Return value: Number of grabbed fields, module 256

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

void Reset_CaptureCounter (unsigned short nDevNo)

Calling this routine sets the half image counter back to zero.

⇔ ❸ Checking for a video signal at the input

short Get_Signal_Status(unsigned short nDevNo);

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

⇔ ❹ Setting the brightness of the image

void Set_Brightness(unsigned short nDevNo,short nBright);

nBright: brightness (-128..127)

⇔ ❹ Reading the brightness setting

short Get_Brightness(unsigned short nDevNo);

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

⇔ ❹ Setting the contrast

**void Set_Contrast(unsigned short nDevNo,
                    unsigned short nContr);**

nContr: contrast (0..511)

⇔ ❹ Reading the contrast setting

unsigned short Get_Contrast(unsigned short nDevNo);

return value: actual contrast value

⇔ ❹ **Setting of the color saturation**

void Set_Saturation(unsigned short nDevNo,
unsigned short nSat_U,
unsignedshort 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)

⇔ ❹ **Reading the content of the registers for color saturation**

unsigned short Get_Sat_U(unsigned short nDevNo);
unsigned short Get_Sat_V(unsigned short nDevNo);

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

⇔ ❹ **Correction of the hue (NTSC only)**

void Set_Hue(unsigned short nDevNo, short nHue);

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

⇔ ❹ **Reading the content of the registers for hue**

short Get_Hue(unsigned short nDevNo);

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

⇔ ❹ **De/activation of the luma-low-pass-filter**

void Set_LDec(unsigned short nDevNo,
unsigned short nOn,
unsigned short nHFilt);

nOn: 1 = Luma decimation on
              0 = Luma decimation off
nHFilt:0   0 = automatic filter selection
              1 = CIF filter
              2 = QCIF filter
              3 = ICON filter

⇔ ❹ De-/Activation of the test image

**void Set_ColorBars(unsigned short nDevNo,
                    unsigned short nColorBars);**

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

⇔ ❹ Adjustment of the range of values

**void LumaControl(unsigned short nDevNo,
                  unsigned short nRange,
                  unsigned short 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

⇔ ❺ Reading/Writing data via the GPIO port

The following three functions control the GPIO port (part of the user connector):

**void Set_GPIO_Direction(unsigned short nDevNo,
                         unsigned short nDirection);**

nDirection: can have values between 0 and 4095

**void Set_GPIO_Data(unsigned short nDevNo,
                    unsigned short nData);**

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

unsigned short Get_GPIO_Data(unsigned short nDevNo);

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

⇔ ❺ ☆ Transmitting Data via the I²C Interface

These functions allow the user to read and write to devices that are connected to the I²C interface.

void I2C_Set_BR_Mode (unsigned short nDevNo, unsigned char bMode)

bMode   Baud rate
               eGrabber-4plus: 0 = 99,2 kHz, 1 = 396,8 kHz
               eGrabber-2:   0 = 33 kHz, 1 = 290 kHz

unsigned char I2C_ReadByte (unsigned char nDevNo, unsigned char bChipAddress, unsigned char bSubAddress, unsigned char *bByteRead)

bChipAddress: Device address of the I²C device connected to the bus
bSubAddress: Internal memory address for the I²C device
*bByteRead: Pointer points to the Byte variable. The result
                is written in the Byte variable.

return value: SUCCESS, NOACK, INVALID_ADDRESS

unsigned char I2C_WriteByte (unsigned short nDevNo, unsigned char bChipAddress, unsigned char bSubAddress, unsigned char bData)

bChipAddress: Device address for the I²C device connected to the bus
bSubAddress: internal memory address for the I²C device
bData: Byte that is written to in the specified address

return Value: SUCCESS, NOACK, INVALID_ADDRESS,
                WRITE_FAILED

⇔ ❺ ☆ Using the Internal EEPROM

The eGrabber-4plus has an internal non-volatile memory available to the user for storing parameters.
The total amount of memory available to the user is 256 Bytes.

Note:
Since the predecessor to the eGrabber-4plus does not have internal memory, the following functions are not compatible.

unsigned char I2C_ReadEEProm (unsigned short nDevNo,
**   unsigned char bSubAddress,**
**   unsigned char *bByteRead)**

bSubAddress: memory address to be read (0x00 … 0xFF)
*bByteRead :   Pointer points to a Byte variable. The result
               is written into the Byte variable.

return value: error code = SUCCESS, NOACK

unsigned char I2C_WriteEEProm (unsigned short nDevNo,
**   unsigned char bSubAddress,**
**   unsigned char bData)**

bSubAddress: Memory address that is to be written to (0x00…0xFF)
bData: Data Byte that is to be written

return value: error code = SUCCESS, NOACK, WRITE_FAILED

⇔ ❺ ☆ Controlling the I/O Pin

The eGrabber-4plus has an external, transistor driven I/O pin. Using control signals, this I/O pin is freely selectable is input or output. Both of the following functions can be used to control the I/O pin:

This function is not available for the eGrabber-2.

void Set_Ext_IO (unsigned short nDevNo, unsigned char bData)

bData: Control value, 0 = CLOSE, 1 – HI-Z

unsigned char Read_Ext_IO (unsigned short nDevNo)

nDevNo   = device number

return value: logical level of the I/O-pin

⇔ ❺ Direct access to the grabber registers

The following functions allow direct access to the registers of the video input controller.

short Read_Local_DWord(unsigned short nDevNo,
                        unsigned short nRegister_Number,
                        unsigned long *lContent);

nRegister_Number: register addressNummer des Registers
lContent: content of the register

short Write_Local_DWord(unsigned short nDevNo,
                         unsigned short nRegister_Number,
                         unsigned long lContent);

nRegister_Number: register address
lContent: content of the register

5.3.4 Program Example DOS

In this section a simple program is described running under DOS, to store an image in the main memory.

For a simple example we restrict us to a constant format of the image and digitize a single shot. The format of the image is 256 x 256 pixels. We assume, that only one eGrabber-4plus is installed in the computer. If more grabbers are installed, only the first Grabber is used.

At first an object „Grabber" is created which provides the interface to the eGrabber-4plus. The image should be stored in an array designated with pEWert. At this point it is necessary to determine the multitude of data for the required image. The dimension of the image is 256 x 256 pixels, and should be stored as RGB-color picture, using the RGB24-format. RGB24 requires 3 Byte per pixel. So we need $256 \cdot 256 \cdot 3$ Byte for one image.

The operation of the Grabber starts with the identification, to check if a Grabber is available. **Max_Device_Number()** delivers the number of Grabber boards installed in the system. If the number = 0, no Grabber board is available in the computer, and the program will be cancelled prematurely.

If one or several Grabber cards are found, the Grabber will be initialized with the function call **Initialize(1)**. The parameter 1 indicates that the first Grabber should be initialized. If several Grabber boards are available the function **Initialize()** must be executed for all Grabber cards.

It makes sense to check after each call with **Get_Error()** if any errors occurred and if the instruction was successful. In order to keep the general view, this is not done in this example.

With the call **Set_Channel(1,1)** channel 1 is selected for the Grabber #1. Digitization will take place for this channel. The following delay is used to synchronize the Grabber to the signal of the camera.

The next function is quite complicated: With the routine **Set_Image(1,...)** the parameter of the image for the Grabber #1 are set. We want to digitize only one even field. Therefore the parameters for the odd field are not required and are set to zero (zero means, that the values in the video processor are not changed). It is important to set the pointer of the odd memory region to NULL, which is an indication to the Grabber, that no odd image has to be produced.

For the parameters of the even image, first the position of the section of the image is defined. The coordinates of the left upper corner are (1.1), so that the section will be found also in the upper left corner of the TV-picture. (**Caution**: (0.0) will leave the values unchanged, which were set before.) As demanded in the problem, we now define the number of pixels required for the resulting image of 256 x 256. The field delivered from the source will be digitized with the complete size of the format with the ratio 4:3. Therefore the size of the image will be set to 344 x 258. A maximum of 360 x 288 would be possible, but 344 x 258 is more suitable because the whole height of the image will be visible in the section. Since the width of the image is different in the right region 88 pixels will be invisible.

With `RGB24` the color format for the even image is defined. Per pixel, three Byte are created, one for each red-, green- and blue value (*see also the instruction Set_Image() and Figure 48*).

As pointer to the image memory region `pEWert` will be provided, which will point to the beginning of the defined array.

For the determination of the video processor setting by the driver, the applied video system must be known. For our example we assume a PAL-camera as image source, and therefore `PAL_BDGHI` is set.

Since we use only one field, *Interlace* = 0 (both fields are stored in separate memories; in addition for our example the odd image is obsolete).

We select the single-shot mode, since only one image should be digitized and not a continuous series.

Now digitization can be started, which is done by invoking the function **Start_Grabber(1)**. During the following while-loop the program waits, until the digitized field will be stored completely in the memory. For this reason we call the function **Data_Present()** for Grabber #1. The returnvalue will be concatenate with 0x04 AND, in order to mask the third bit (bit # 2). This is the *S_EVEN*-bit (*see Figure 49*). This bit will be set, after the even image is stored completely in the memory. Since only this information is interesting to us, the other status bits are masked out. The delay time in the loop prevents, that the status is permanently inquired, which might be disturbing for the data transfer on the PCI-Bus.

After the end of the digitization, the instruction **Stop_Grabber()** has to be given, so that the Grabber is set to the none operative status. Otherwise the Grabber remains in the standby position and no further digitization can take place.

The image is now found in the defined array and can be evaluated. In order to repeat recording of a new image with the same dimensions from the same channel, the program can call again the function **Start_Grabber()**.

```
#include <dos.h>
#include <malloc.h>
#include <stdio.h>
#include <stdlib.h>
#include "pci4grab.h"


PCI_GRABBER4 Grabber; // Object for the access to the Grabber
static unsigned char pEWert[256*256*3]; // Memory for EVEN-image

main ()
{
    if((Grabber.Max_Device_Number())==0)
    {
     printf("No eGrabber-4plus was found!!") ;
     exit(1) ;
    }
    Grabber.Initialize(1) ;     // First initialize Grabber!
    Grabber.Set_Channel(1,1) ;  // Select input channel and
    delay(100);                 // Wait, until synchronized
    Grabber.Set_Image( 1,       //Grabber-number
                      0,0,      //All ODD-parameters are...
                      0,0,      //...negligible and ...
                      0,0,      //...therefore...
                      0,        //...are set to 0 .
                      NULL,     //Pointer to ODD-image=NULL
                                //=> no ODD-image
                      1, 1,     //upper left corner of the EVEN-
                                 image
                      256,256,  // Size of the image section
                      344,258,  // Size of the image
                      RGB24,    // Color information: RGB24-image
                      pEWert,   // Pointer to EVEN-image memory
                      PAL_BDGHI,// Color system
                      0,        // Interlaced-modus off
                      1);       // Single shot on

    Grabber.Start_Grabber(1);   // Start grabbing
    while(!(Grabber.Data_Present(1)&0x4))
          delay(10);            // Wait until data are available
    Grabber.Stop_Grabber(1);    // Stop grabbing
// ** Image is stored in the memory and can be evaluated **
}
```

6 Changes to the eGrabber-2 and Compatibility

The eGrabber-4plus is a successor to the eGrabber-2. This means, by design, that these models possess downwards compatability. That means:

- **Windows Applications:**

  The eGrabber-4plus can replace any eGrabber-2s that have previously been installed in systems. Under the Windows operating system, there are no changes for existing application software.
  Although, a new device driver must be installed for the eGrabber-4plus.
  The eGrabber-4plus offers the same functionality as the eGrabber-2. Additional functions for the eGrabber-4plus cannot be used with older generation application software for the eGrabber-2.

- **DOS Applications:**

  Under DOS applications a separate device driver is not used, rather the driver is linked as a library for the compiler in user programs. In order to ensure that the eGrabber-4plus operates with these programs, a new driver library must be linked. This means that the program must be newly compiled. There are no changes for user specific program code. The software interfaces for the driver library under the eGrabber-4plus and the eGrabber-2 are also compatible.

- **Old Grabber and New Applications**

  It is possible to install software that has been developed for the eGrabber-4plus into a system populated with the eGrabber-2 (EPC-032) and subsequently use the new driver.
  This is possible because the new driver supports older card versions. Although, please note, that many hardware applications are no longer available.
  If functions or parameters are used that the older Grabber does not support, the driver sends an error code to the user's program.
  The functions *Read_GrabberInfo* and *Read_OrderCode* can be used to determine which Grabbers and features are available.

- **Hardware Compatibility**

  All of the signals available on the eGrabber-2 are also available at the same sockets and contacts on the eGrabber-4plus. Connector cables previously used for the eGrabber-2 can also be used for the eGrabber-4plus.

  With the eGrabber-4plus some signals have been added to **pin header row X3**. The pin header row therefore has 6 additional pins when compared to that of the eGrabber-2. If you should choose to implement the eGrabber-4plus in an existing application as a substitution for an eGrabber-2, you can simply remove the last 6 pins from the pin header row X3. The first 20 pins are identical to those of the eGrabber-2 and you can power it with the same power adapter.
  In order to make it easier for you to remove the pins, those pis that are compatible with the eGrabber-2 have a white border around them.

  One of the main differences between the eGrabber-4plus and the eGrabber-2 is that the SMB-Sockets P3 and P4 have been removed. Previously it was possible to feed in an S-Video signal via these sockets. This function is now taken over by the Mini-DIN-Socket X4

  Two pins have been altered on the expansion connector (Option Port, X6) (*see Table 14*).

Option Port, X6					
Pin	**Function**	**Pin**	**Function**	**Pin**	**Function**
1	+5V out	8	I/O 6	**15**	**I/O-Pin**
2	I/O0	9	I/O 7	16	I/O Clk
3	I/O1	10	I/O 8	17	I²C SCL
4	I/O2	11	I/O 9	18	I²C SDA
5	I/O3	12	I/O 10	19	GND
6	I/O4	13	I/O 11	20	GND
7	I/O5	**14**	**N.C.**	bold = new function	

Table 14:   Pin Assignment of the Option Port - Connectors (Both Models)

- **Power Supply to the Camera**

  In order to supply an operating voltage to connected cameras, a direct current can be drawn from the pin header row. This is directly connected to socket X300 or, depending on Jumper JP301, drawn from the PC/104 connector or the integrated supply voltage. This function was not available with the eGrabber-2.

- **Changing the Old Driver Version (for V3.0)**

  If software has been developed for the eGrabber-4 that is still used in conjunction with older driver versions (before Version 3.0), please read the following notes:

  - The Window's DLL was expanded with the function **GetVersionNumber**. If a non-valid function pointer is returned with **GetProcAddress**, then a newer version of **GR4CDLL.DLL** is required.
  - **Set_Image Function**:

    In previous versions, the *hpos* and *vpos* parameters in the Window's DLL were required to be set to 1 in order to place the cropped image in the upper left-hand corner of the digitized image. Now the parameters accept only even values, this means that *hpos* and *vpos* must be set to 0.

    L-608e_4

7 Trouble-Shooting

If your Grabber does not function as expected, you can eliminate the occurrence of minor glitches by going through the following check list. If, after going through the check list, you are unable to eliminate the error on your own, please consult PHYTEC-Support.

❍ 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 eGrabber-4plus, 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?

  ● Are both S-Video inputs connected (only one's possible)?

  ● Was the eGrabber-4plus 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 distur-bances.
  ● 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 computer stalls under DOS during the start of the program/ an error message appears
  ● Another device driver or the memory management for example EMM386, might cause trouble. Remove the installation of those components or use other drivers.

○ 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!

○ 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()* .

❍ For the DOS-demo program only a black/white image is displayed (the first two options in the menu are missing).
  ● The DOS-program requires for the presentation of color pictures and for pictures with higher resolution, a graphic card equipped with a graphic processor ET4000 / ET6000 (Tseng Labs). The program example under DOS, does not take into consideration all the peculiarities of the various graphic cards. For the presentation with true colors for a resolution of 640 x 480 pixels the standard-VGA-Modus is not sufficient.

❍ 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 integrated supply voltage does not function and the green LEDs fail to illuminate.

● Check the model number, is it an EPC-032-X2 or an EPC032-X1-X2? The adapter is only present with these models.

● Is the input voltage connected (8 to 25 V / 15 to 25 VDC)?

● Is the polarity of the input voltage correct?

● Is the plug correctly connected to X300?

● Is the circuit breaker SI301 functioning?

● Has a short circuit / overload occurred (adapter shuts off)?

● Was the adapter switched off over connector X301? For operation X301 must be open or Pin 1 has to be on HIGH level. If necessary, as a test, remove one of the connectors from X301.

○ The green LED for +12 V does not illuminate and the 12 V output voltage is not present. The +5 V supply voltage functions.

● The input voltage is below +15 VDC. At least a 15 V input voltage is required for operation of the 12 V branch of the adapter.

● A short circuit has occurred in the 12 V branch or the current draw is too high.

❍ The integrated supply voltage does not function although the green LEDs illuminate.

  ● Using a multimeter check the voltages on X302

  ● The supply voltage is not connected to the Grabber or the CPU-card. You will have to establish this connection yourself. Normally this can be achieved via a cable connection to X302. Read the manual for the CPU-card to find out where the optimal location for connecting the supply voltage is.

  ● If the LEDs only illuminate weakly, it is possible that too much current is being drawn from the adapter. The overload protection then automatically reduces the output voltage in order to prevent damage to the adapter.

  ● If you wish to direct the voltage into the system over the PC/104 headers, you have to configure the jumper field JP301 in a corresponding manner (*see section 1.10.3*)

  ● If you use a Jumptec MOPS-CPU-Card, you may have to populate the socket connector X302 in order to supply the CPU-card with voltage without using a cable.

❍ The socket connector X302 for the CPU supply voltage is not present.

  ● The connecting socket is usually not populated, since it is only suitable for cards from certain manufacturers (for example Jumptec, MOPS-Series). In order to prevent conflicts with other components or connectors from incompatible cards, the PHYTEC connector is not soldered on

❍ When switched on the system does not initialize properly or the green control LEDs flicker.

- The power-on current of connected components (i.e . hard drives, CD-ROM drives) is too high. This activates the overload protection circuitry of the adapter. We recommend using components that were developed for laptops (for example 2.5" hard drives). If absolutely necessary you can attempt to reduce the power-on current by adding resistance to the device's supply voltage (0.5 to 2Ω).

❍ 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
- 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.

❍ How is it possible to use several eGrabber-4plus at the same time with Windows'95?

● Under Windows'95 all eGrabber-4plus use the same memory portion of the main memory of the PC. So only one digitization process can be executed at the time. But during digitization of one Grabber the other grabbers can respond and parameters can be written or read or be changed. This is an advantage for example if the channel selection should be changed and the time span could be used as lead time, so that the analogue signal is recognized correctly by the Grabber. Under DOS the programmer can define a separate memory region for each Grabber. Here parallel processing is possible. However it must be considered, that the transfer rate of the PCI-bus is not exceeded.

Under DOS or Windows 98/2000/NT/ME/XP each Grabber can utilize a separate memory space for image storage

Index

Document:	eGrabber-4*plus*
Document number:	L-608e_4, June 2004

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