Referenz Handbuch

Inspecta - Software - Level 1

Für Inspecta 3 und Inspecta 4

Version 1.0.8

Mikrotron GmbH D-85716 Unterschleißheim Tel.++49 89 726342 00 Fax.++49 89 736342 99 www.mikrotron.de

1 ALLGEMEINES 4

1.1 Über Level1 4

1.2 Aufbau des Handbuches 4

1.3 Revisionsgeschichte 4

1.4 Markenrechte 5

2 INSTALLATION 6

2.1 Setupdiskette / CD 6

2.2 Installationshinweise für Windows® NT/2000/XP 6
2.2.1 Definition des Bildspeichers, compatibility mode 6
2.2.2 Definition des Bildspeichers, „Maxmem“ mode 6
2.2.3 Registry Einträge 6
2.2.4 INSPECTA Installation unter Windows® NT 7
2.2.4.1 Mehrere Inspectas in einem PC unter WinNT 7
2.2.5 INSPECTA Neuinstallation unter Windows® 2000/XP 8
2.2.6 INSPECTA Treiberupdate unter Windows® 2000/XP 8
2.2.6.1 Mehrere Inspectas in einem PC unter Win2000/XP 9

3 FUNKTIONSBESCHREIBUNGEN 10

3.1 mvfg_open (pcCameraProfile, DeviceNumber) 10

3.2 Parameter setzen und lesen 11
3.2.1 mvfg_setparam (pcParamName, pcParamValue, DeviceNumber) 11
3.2.2 mvfg_getparam (pcParamName, pValueBuffer, DeviceNumber) 12
3.2.3 Parameter von get_/set_param() 13

3.3 mvfg_getbufptr(DeviceNumber) 20

3.4 mvfg_grab(iCommand, DeviceNumber) 21

3.5 mvfg_close(DeviceNumber) 23

3.6 mvfg_errmessage(iCode) 24
3.6.1 Beispiele 25

3.7 Öffnen und Schließen des Treibers 26

3.8 Setzen und Lesen von Parametern 27

3.9 Bildeinzug und Maße des Bildes 28

4 BEGRIFFSDEFINITIONEN 30

4.1 Black Lines 30

4.2 Frames 30

4.3 Inspecta-XX.cam 30

4.4 Planes 32

4.5 Speicherformate 33
4.5.1 8 Bit B&W 33
4.5.2 8 Bit B&W - interlaced 33
4.5.3 8 Bit B&W - interlaced - 2 Halbbilder 34
4.5.4 8 Bit B&W – Inhalt zweier Zeilen als gerade und ungerade Pixel 34
4.5.5 8 Bit B&W – erste Hälfte einer Zeile als gerade Pixel, zweite Hälfte einer Zeile als ungerade Pixel 35
4.5.6 8 Bit B&W – erste Hälfte einer Zeile als gerade Pixel, zweite Hälfte einer Zeile als ungerade Pixel in umgekehrter Reihenfolge 35
4.5.7 8 Bit B&W - n Kameras - n-planes 36
4.5.8 8 Bit B&W - n Kameras - packed 36
4.5.9 10 Bit B&W 37
4.5.10 10 Bit B&W – erste Hälfte einer Zeile als gerade Pixel, zweite Hälfte einer Zeile als ungerade Pixel 37
4.5.11 10 Bit B&W – erste Hälfte einer Zeile als gerade Pixel, zweite Hälfte einer Zeile als ungerade Pixel in umgekehrter Reihenfolge 37
4.5.12 16 Bit RGB 38
4.5.13 24 Bit RGB - packed 38
4.5.14 24 Bit RGB - 3-planes 39
4.5.15 32 Bit xRGB 39

4.6 Testmode 40

4.7 VSYNC-Signal 40

1 Allgemeines

1.1 Über Level1

Mit dem Level1 API sind nur wenige Funktionsaufrufe nötig um den Inspecta zu initialisieren, eine bestimmte Kamera auszuwählen und ein Bild einzuziehen.

Level1-Funktionen wurden entwickelt, um die Programmierung des Inspecta zu erleichtern und zu beschleunigen.

Dieses Handbuch beschreibt die Level1 Funktionen für die Framegrabber der Serie Inspecta-3 und Inspecta-4.

	**Level1 Funktionen sind nur in WinNT/2k/XP implementiert. Für die Programmierung von Win9x-Programmen muss auf Level0 Funktionen zurückgegriffen werden.**.
	**Verwenden Sie eine Treiberversion ab 3.28.**

1.2 Aufbau des Handbuches

Dieses Handbuch ist für den versierten Anwendungsprogrammierer geschrieben. Es beschreibt die Benutzung der Level1-Funktionen unter WinNT/2k/XP.

Für die detaillierte Beschreibung der Funktion und der Hardware des Inspecta ist auf das Hardware Manual hingewiesen.

Funktionsbeschreibungen:
Die Funktionsbeschreibungen sind immer auf mehrere Seiten aufgeteilt. Auf der ersten Seite befindet sich immer ein Hinweis auf ein Beispiel, welches sich im Abschnitt "Beispiele" befindet.

Begriffsdefinitionen:
Eine Reihe von Begriffen und Speicherformaten werden erklärt, die zum Verständnis des Handbuches hilfreich sind. Bei Verwendung der Online-Version des Handbuches befinden sich im Text Hyperlinks zu den Begriffsdefinitionen.

1.3 Revisionsgeschichte

Dieses Handbuch ist mit großer Sorgfalt erstellt worden. Wir können trotzdem Fehler nicht ausschließen. Wir haften nicht für die Folgen solcher Fehler. Wir entwickeln den Inspecta ständig weiter. Dieses Handbuch unterliegt aber keinem Änderungsdienst.

Ab Treiberversion 3.46: zusätzliche Parameter-Namen für die Funktionen
_mvfg_setparam(), _mvfg_getparam_()
MVFGPAR_TRIGGERMODE
MVFGPAR_CAPFRAME

Ab Treiberversion 3.82: zusätzlicher Parameter-Name für die Funktionen
_mvfg_getparam_():
MVFGPAR_FORMAT_IDENT

1.4 Markenrechte

Microsoft , Windows, Windows 95 Windows NT, Windows 2000, Windows XP sind geschützte Markenzeichen von Microsoft.
Intel, das Intel Inside Logo, Pentium® sind eingetragene Marken von Intel Corporation in den USA und anderen Ländern und werden unter Lizenz genutzt.

2 Installation

2.1 Setupdiskette / CD

Zum Lieferumfang des Inspecta Frame Grabber gehört eine Setupdiskette oder CD.
Die neueste Treiberversion ist auch unter http://www.mikrotron.de abrufbar.

Auf der CD oder unter http://www.mikrotron.de/ finden Sie die Manuals:
Inspsd.pdf Inspecta Softwarebeschreibung
Insphd.pdf Inspecta Hardwarebeschreibung

2.2 Installationshinweise für Windows® NT/2000/XP

2.2.1 Definition des Bildspeichers, compatibility mode

Die Definition der Bildspeichers erfolgt ab Treiberversion 2.27 bei Neuinstallationen im sog. „compatibility mode“. Dabei wird ein Bildspeicher von 8MB für jeden INSPECTA aus dem „NonPagedMemoryPool“ reserviert. Ein Registry Eintrag zeigt an, wieviel Bildspeicher maximal zur Verfügung steht.

Bei diesem Verfahren ist händischer Eingriff in eine Systemdatei nur nötig, wenn größerer Bildspeicher als 8Mb benötigt wird. (Siehe Registry Einträge).

2.2.2 Definition des Bildspeichers, „Maxmem“ mode

Die Definition der Bildspeichers erfolgt über den MAXMEM Schalter im BOOT.INI File. MAXMEM begrenzt den Speicher der dem System zugewiesen ist auf den im Schalter angegebenen Wert:

```
/MAXMEM=32 ; Das System erhält maximal 32 MB Hautpspeicher zugewiesen.
```

Darüberhinaus existierender Speicher steht als Bildspeicher für die den INSPECTA benützende Applikation zur Verfügung. Der Gesamtspeicher muß bekannt und um mindestens 1MB größer als MAXMEM sein.

Die BOOT.INI Datei ist versteckt und schreibgeschützt. Sie kann in der DOS Box entsperrt werden:

```
cd \
attrib boot.ini -r -h -s
```

Danach kann die Datei editiert, und der Switch: /MAXMEM=xxx an das Ende der für WinNT vorgesehenen Zeile angefügt werden.

2.2.3 Registry Einträge

Folgende Schlüssel werden verwendet:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\MpfgNt

Sie haben folgende Werte (Beispiel):

Name	Wert	Bedeutung
BlockSize	0x00800000	*Länge des Bildspeichers in Byte (Beispiel:8MB)*
MemStartPage	0x02000000 0	*bei MAXMEM: Bildspeicher Start-adresse (Beispiel: 32 MB)* *bei compatibility mode*
MemoryAllocationMode	0 1	*compatibility mode* *MAXMEM mode*
Devicenumber	0x00000000	*für ersten INSPECTA, benütze 1,2, und 3 für bis zu vier INSPECTA*

Keine anderen Werte ändern!

2.2.4 INSPECTA Installation unter Windows® NT

Es ist die Version: **Workstation 4.0 mit eingespieltem Service Pack, mindestens 3,** notwendig.
Die Installation erfolgt von der mitgelieferten Setup Diskette in folgende Verzeichnisse:

%WINBOOTDIR%\system32\driver\mpfgnt.sys: Windows NT device driver
%WINBOOTDIR%\system32\mvfgd32.dll: Windows NT dynamic link library

Die mvfgd32.dll hat den gleichen Namen und beinhaltet die gleichen Funktionsaufrufe wie die für Windows9x verwendeten, ist jedoch ein anderer Binärfile und darf nicht mit der mvfgd32.dll von Windows9x verwechselt werden. Die Namensgleichheit erlaubt die Verwendung von Apllikationen ohne Änderung oder Neucompilation unter den verschiedenen Windows® Betriebssystemen.

Die Hilfsprogramme (VCAM95) und Beispiele werden wenn nicht anders angegben nach:

..\Programme\Mikrotron GmbH\Inspecta

installiert.

2.2.4.1 Mehrere Inspectas in einem PC unter WinNT

Für vier möglich Inspectas werden schon bei der Installation eines einzigen Inspectas vier Registry Schlüssel und vier Treiber angelegt.
Die Zahl der installierten Inspecta wird bei der Installation angegeben. Um nachträglich mehrere Inspecta zu installieren, kann das Control-Applet "Inspecta" in der Systemsteuerung. verwendet werden.
Um den reservierten Speicher für die einzelnen Inspectas einzustellen, kann dieses Applet auch verwendet werden.

2.2.5 INSPECTA Neuinstallation unter Windows® 2000/XP

Wenn zum ersten mal ein Inspecta ins System ein- oder hinzugefügt wird, wird die neue Hardware vom Plug & Play Manager erkannt. Um die Treiberinstallationsprozedur erfolgreich abzuschließen müssen mindestens die Files:

mpfg.inf und mpfgnt.sys

vom Installationsmedium vorhanden sein. Diese sollten also auf einem Wechseldatenträger vorhanden, oder auf die Festplatte des Systems kopiert sein.

Danach müssen die Treiber aktiviert werden:

Treiber aktualisieren -> weiter -> Treiber aus Liste wählen -> weiter -> Datenträger -> Durchsuchen -> auf Inspecta Treiber Medium zeigen -> mpfg.inf wählen -> öffnen -> o.K. -> weiter -> Warnhinweis ignorieren -> weiter -> Fertig stellen -> PC runterfahren -> Ausschalten.

Wenn vor dem Wiedereinschalten ein Inspecta gewechselt oder die Anzahl oder der Steckplatz geändert wurde, muß in der Regel nochmals gebootet werden.

2.2.6 INSPECTA Treiberupdate unter Windows® 2000/XP

Dazu wird vom Installationsmedium mittels „Setup“ installiert. Damit werden die neueste Version der Hilfsprogramme, Kameradateien und Beispiele installiert. Alternativ müssen mindestens die Files:

mpfg.inf und mpfgnt.sys

vom Installationsmedium vorhanden sein. Diese sollten also auf einem Wechseldatenträger vorhanden, oder auf die Festplatte des Systems kopiert sein.

Danach müssen die Treiber aktiviert werden:

Systemsteuerung -> System -> Geräte Manager -> Audio-, Video und Gamecontroller -> Inspecta Framegrabber -> rechts click -> Eigenschaften -> Treiber -> Treiber aktualisieren -> weiter -> Treiber aus Liste wählen -> weiter -> Datenträger -> Durchsuchen -> auf Inspecta Treiber Medium zeigen -> mpfg.inf wählen -> öffnen -> o.K. -> weiter -> Warnhinweis ignorieren -> weiter -> Fertig stellen -> PC runterfahren -> Ausschalten.

Wenn vor dem Wiedereinschalten ein Inspecta gewechselt oder die Anzahl oder der Steckplatz geändert wurde, muß in der Regel nochmals gebootet werden.

2.2.6.1 Mehrere Inspectas in einem PC unter Win2000/XP

Nach der Installation der Treiber und dem obligatorischen Neustart übernimmt der Plug und Play Manger die Aktivierung der Treiber und die Numerierung der Device-Ids der eingesteckten Inspectas.

Bei jeder Änderung der Anzahl oder des Steckplatzes der eingesteckten Inspectas muss ein zusätzlicher Neustart vorgenommen werden. Die Anzahl der zu aktivierenden Inspectas bestimmt der Plug und Play Manager automatisch.

3 Funktionsbeschreibungen

3.1 mvfg_open (pcCameraProfile, DeviceNumber)

Synopsis:

```
LONG WINAPI mvfg_open( char * pcCameraProfile, LONG DeviceNumber )
```

Beschreibung:
Der ausgewählte Inspecta (einer von bis zu vieren bei WinNT) wird zusammen mit einer Kamera aus der selektierten Kameradatei mit der dazugehörigen Sektion initialisiert.
Wenn in der Kameradatei ein Hinweis auf eine Kamerastring-Datei vorhanden ist, werden die Bytes aus dieser Datei über die serielle Schnittstelle des Inspecta 4D/4C an die Kamera geschickt.
Es wird empfohlen, die Konfigurationsdatei Inspecta-XX.cam (z.B.: Inspecta-4A.cam), die vom interaktiven Kamera-Einstellungsprogramm VCAM95.EXE geschrieben wird, zu verwenden.

Beispiel:
siehe Beispiel Öffnen und Schließen des Treibers

Parameter:

*char * pcCameraProfile*
pcCameraProfile zeigt auf einen String, der den Namen der Kamera-Konfigurationsdatei und der spezifischen Kamera-Sektion enthält. (z.B. "Inspecta-4A.cam;TestMode"). Dateiname und Sektionsname sind durch ";" getrennt.
Eine Sektion in der Konfigurationsdatei ist gekennzeichnet durch "[Sektionsname]". Unterhalb der Sektionskennzeichnung sind alle nötigen Kameraeinstellungen angegeben, welche von mvfg_open automatisch übernommen werden.

LONG DeviceNumber
Grabber-Nummer (0 bis 3)

Funktionswert (LONG):

MVFG_OK
Initialisierung erfolgreich

EMVFG_NO_VXD
Fehler: Der Grabber ist nicht installiert

EMVFG_CAMFILE_NOTFOUND
Fehler: Konfigurationsdatei existiert nicht

EMVFG_CAMSECTION_NOTFOUND
Fehler: Sektion in der Konfigurationsdatei existiert nicht

EMVFG_CAMSTRG_FILE_NOTFOUND
Fehler: Kamerastring-Datei angegeben, aber nicht gefunden

Sonstige
allgemeiner Fehler

3.2 Parameter setzen und lesen

Mit der Funktion mvfg_setparam() können spezifische Inspecta-Parameter gesetzt und mit mvfg_getparam() wieder zurückgelesen werden. Die unterschiedlichen Parameter sind in der Tabelle unter Kapitel 3.2.3 angegeben. 3 Parameter können nur gelesen werden:

- MVFGPAR_CAPFRAME,
- MVFGPAR_FORMAT_INFO
- MVFGPAR_FORMAT_IDENT

3.2.1 mvfg_setparam (pcParamName, pcParamValue, DeviceNumber)

Synopsis:

```
LONG WINAPI mvfg_setparam( char * pcParamName,
                           char * pcParamValue,
                           LONG   DeviceNumber )
```

Beschreibung:
Spezifische Inspecta-Parameter können neu gesetzt werden, auch wenn sie zuvor schon durch mvfg_open gesetzt wurden.

Beispiel:
siehe Beispiel Setzen und Lesen von Parametern
Parameter:

*char * pcParamName*
Name des Parameters, der geändert werden soll.
(Entsprechende Konstanten sind in der Tabelle auf Seite 13 aufgeführt.)

*char * pcParamValue*
Einzustellender Wert des Parameters.
(Entsprechende Wertebereiche sind ebenfalls in der Tabelle auf Seite 13 aufgeführt.)

LONG DeviceNumber
Grabber-Nummer (0 bis 3)

Funktionswert:

MVFG_OK
Der gewählte Parameter wurde gesetzt.

EMVFG_NO_VXD
Fehler: Der Grabber ist nicht installiert.

EMVFG_CAMPARAM_UNKNOWN
Fehler: Der Parameter ist unbekannt.

EMVFG_CAMPARAM_BADVALUE
Fehler: Der Wert für den Parameter ist unzulässig.

EMVFG_NOT_OPEN
Fehler: Der Treiber wurde nicht mit mvfg_open geöffnet oder er wurde schon wieder geschlossen.

3.2.2 mvfg_getparam (pcParamName, pValueBuffer, DeviceNumber)

Synopsis:

```
LONG WINAPI mvfg_getparam( char * pcParamName,
                           void * pValueBuffer,
                           LONG   DeviceNumber )
```

Beschreibung:

Die Funktion liest einen Parameter, der durch mvfg_open oder mvfg_setparam gesetzt wurde.

Beispiel:
siehe Beispiel Setzen und Lesen von Parametern

Parameter:

*char * pcParamName*
Name des Parameters, der gelesen werden soll.
(Entsprechende Konstanten sind in der Tabelle auf der nächsten Seite aufgeführt.)

*void * pValueBuffer*

Adresse einer Variablen, in der der ausgelesene Wert geschrieben werden soll.
Alle Variablen müssen den Typ des Parameters haben (siehe Beispiel).

LONG DeviceNumber
Grabber-Nummer (0 bis 3)

Funktionswert:

MVFG_OK
Der gewählte Parameter wurde gelesen.
EMVFG_NO_VXD
Fehler: Der Grabber ist nicht installiert.
EMVFG_CAMPARAM_UNKNOWN
Fehler: Der Parameter ist unbekannt.
EMVFG_NOT_OPEN
Fehler: Der Treiber wurde nicht mit mvfg_open geöffnet oder er wurde schon wieder geschlossen.

Der Puffer pValueBuffer muss vom richtigen Typ sein. Der Parameter MVFGPAR_FORMAT_INFO ist vom Typ FORMAT_INFO, alle anderen Parameter sind vom Typ long

3.2.3 Parameter von get_/set_param()

pcParamName	pcParamValue	
MVFGPAR_CAMMODE	Kameramodus, d.h. eine Kennziffer für eine bestimmte Kamera, die verwendet werden soll, als dezimale Kennziffer (z.B. „81“)	
MVFGPAR_LINELEN	Länge einer Bildzeile für Aufnahme in Bytes (z.B. „740“)	
MVFGPAR_NUMLIN	Anzahl der Bildzeilen für Aufnahme (z.B. „602“)	
MVFGPAR_BLACKLINESBEGIN	Anzahl zu verwerfender Bildzeilen oben (z.B. „26“)	
MVFGPAR_BLACKLINESEND	Anzahl zu verwerfender Bildzeilen unten (z.B. „10“)	
MVFGPAR_BLANKTIME	Anzahl der Pixel ohne Bilddaten zwischen zwei Zeilen (z.B. „85“)	
MVFGPAR_WHITELEVEL	Weißschwelle („0“ ... „255“)	
MVFGPAR_BLACKLEVEL	Schwarzschwelle („0“ ... „255“)	
MVFGPAR_VIDEONORM	Videonorm	
	„0“	NTSC/SECAM
	„1“	PAL/CCIR
MVFGPAR_INTERLACED	Interlaced	
	„0“	Non-interlaced Kamera
	„1“	Interlaced Kamera
MVFGPAR_REQFRAME	Anzahl / Position der zu grabbenden Frames	
	„0“ ... „n“	an (n + 1). Position einen Frame schreiben
	„-2“ ... „-n“	ab 1. Position (n – 1) Frames schreiben
	„-1“	ab 1. Position so viele Frames wie im Speicher platz haben.
MVFGPAR_CAMSEL	Auswahl Kameraeingang („0“ .. „7“)	
MVFGPAR_DIGSEL	Kameranummer der Kamera, die über einen Multiplexer ausgewählt wird	
MVFGPAR_CAMPORT	Auswahl eines Kameraeingangs, unabhängig von der Inspecta-Version: - Bei Analog-Grabbern (Inspecta4A) wird der Inspecta-interne Multiplexer angesteuert („0“ ... „7“), - bei digitalen Grabbern (Inspecta4D) wird der externe MUX5000 angesteuert („0“ .... max. „15“) und - bei Camera Link Grabbern (Inspecta4C) wird der B563 Camera Link Multiplexer gesteuert („0“ ... „2“)	
MVFGPAR_IOPORT	Lesen und Setzen der 4 digitalen Ein- und Ausgänge; gesetzt und gelesen können Bit 0 bis Bit 3, d.h. „0“ ... „15“	
MVFGPAR_PHOTO	Belichtungszeit bei geshutterten Kameras (z.B. „30“). Die Kamera muss sich dabei im asynchronen Modus befinden.	
MVFGPAR_PHOTOFLAG	Shuttern durch Grabber	
	„0“	disabled
	„1“	enabled

pcParamName	pcParamValue	
MVFGPAR_SCANPERIOD	Zeilenfrequenz einer Zeilenkamera, nur für Inspecta 2/3 und 4A	
MVFGPAR_CONTINUOUSFLAG	Legt fest, ob der Frame Grabber im kontinuierlichen oder im start/stopp Modus betrieben wird. Siehe Parameter MVFGPAR_PHOTOFLAG und MVFGPAR_EXTPHOTOFLAG	
	"0":	Stopt den Frame Grabber nach einer Aufnahme. Das Flag muss bei asynchronem Betrieb der Kamera auf "0" gesetzt werden! Asynchroner Betrieb liegt vor, wenn die Kamera durch ein extern zugeführtes Signal zur Aufnahme eines Bildes angeregt wird.
	"1":	Kontinuierlicher Bildeinzug. Bei kontinuierlichem (synchronem) Betrieb der Kamera, muss das Flag auf "1" gesetzt werden! Ein synchroner Betrieb der Kamera liegt vor, wenn die Kamera selbständig, d.h. ohne ein externes Signal, kontinuierlich Bilder liefert.
MVFGPAR_EXTPHOTOFLAG	Selbständige, asynchrone Auslösung des Belichtungsvorgangs durch externes Signal (Trigger).	
	„0“	Externes Signal ausschalten
	„1“	Externes Signal zulassen.
MVFGPAR_TRIGGERMODE	Spezifiziert die Art des Bildeinzugs bei Verwendung des externen Signals (Triggers):	
	"0"	mit der Flanke des Signals Auslösen des Belichtungsvorgangs
	"5"	solange das Signal aktiv ist, erfolgt Bildeinzug (für Zeilenkameras)
MVFGPAR_BAUDRATE	Baudrate des seriellen Kanals bei Camera Link frame grabbern, bspw „9600“, „19200“, ...	
MVFGPAR_TIMEOUT	Timeout in ms	

pcParamName	pcParamValue		
MVFGPAR_ENAENC	Enable encoder		**Nur für Zeilenkameras**
	"0"	Encoder ist inaktiv	
	"1"	einphasiges Encodersignal auf Bit 2 des opto Eingangs geteilt durch den Teiler: *MVFGPAR_DIVIDER* steuert das Auslösen einer Zeile unabhänging von der Drehrichtung des Encoders.	
	"2"	zweiphasiges Encodersignal auf Bit 1 und 2 des opto Eingangs geteilt durch den Teiler: *MVFGPAR_DIVIDER* steuert das Auslösen einer Zeile in Vorwärstrichtung des Encoders. In Rückwärtsrichtung werden die Drehimpulse gezählt, jedoch erfolgt kein Auslösen einer Zeile. Es müssen erst soviele Vorwärtsimpulse eintreffen wie vorher Rückwärtsimpulse gezählt wurden, bevor Zeilen erneut ausgelöst werden. Die Richtung wird durch vertauschen der beiden Ausgänge des Encoders an den Eingängen Bit 1 und 2 bestimmt.	
	"3"	zweiphasiges Encodersignal auf Bit 1 und 2 des opto Eingangs geteilt durch den Teiler: *MVFGPAR_DIVIDER* steuert das Auslösen einer Zeile in Vorwärstrichtung des Encoders. In Rückwärtsrichtung werden keine Drehimpulse gezählt. Die Richtung wird durch vertauschen der beiden Ausgänge des Encoders an den Eingängen Bit 1 und 2 bestimmt.	
	Für MVFGPAR_ENAENC = “1”, “2”, oder “3”: Bei Stillstand des Transports werden nach einer Zeit: Pixeltakt/65535 Zeilenimpulse ausgelöst, die Zeilendaten jedoch nicht übernommen. Trifft während eines solchen Leerzyklus wieder ein Encoderimpuls ein, wird dieser nach Beendigung des Leerimpulses nachgeholt. Dieses Verfahren vermeidet, dass die ersten Zeilen nach Stillstand überbelichtet werden		
MVFGPAR_DIVIDER	teilt den Encodertakt durch diesen Wert und löst je nach Richtung eine Zeile in der Kamera aus. Der Teilerfaktor ist 1+*MVFGPAR_DIVIDER* für *MVFGPAR_DIVIDER* = 0...255.		
MVFGPAR_SCANRATE	teilt den Pixeltakt der Kamera durch diesen Wert und steuert damit die Horizontalfrequenz der Zeilenkamera, solange MVFGPAR_ENAENC = 0 ist. Der Wert muß größer als die Zeilenlänge der Kamera, und kleiner als 65535 sein		
MVFGPAR_EXPTIME	Belichtungszeit (benötigt: Treiberversion >= 2.75 und Inspecta-4D/C Hardware). Für die Belichtungszeit (SZ) bei den verschiedenen Typen der Inspecta Serie gilt:		
	Inspecta-2/3	Keine SZ-Kontrolle möglich	
	Inspecta 4D	SZ=1/Pixelclock* MVFGPAR_EXPTIME $SZ_{max.}$=1/Pixelclock* 65530	
	Inspecta 4C	SZ= 1µs * MVFGPAR_EXPTIME $SZ_{min.}$= 2µs $SZ_{max.}$= 8192µs	

pcParamName	pcParamValue	
Hinweis für Zeilenkameras	Für gleichbleibende Helligkeit des Bildes bei unterschiedlicher Transportgeschwindigkeit muss die Zeilenkamera mit einer Belichtungssteuerung ausgerüstet sein. Bei einer Belichtungszeit, die durch die Zeilenkamera bestimmt ist (je nach Kamerahersteller z.B.: „programmable exposure time“), wird der Parameter *MVFGPAR_EXPTIME* auf die für die Kamera kürzestmögliche Pulszeit gestellt. (In der Regel: 80). Die maximale Zeilenfrequenz ergibt sich dann aus: 1/(Belichtungszeit+Auslesezeit) Bei einer Belichtungszeit, die durch den Inspecta-4D/C bestimmt ist (je nach Kamerahersteller z.B.: „level controlled exposure time“), wird der Parameter *MVFGPAR_EXPTIME* auf die gewünschte Länge gestellt. Die maximale Zeilenfrequenz = 1/(Auslesezeit) kann erreicht werden, wenn die eingestellte Belichtungszeit kleiner als die Auslesezeit ist. Wenn die Bilder einzeln getriggert werden, (externes Triggersignal auf Opto-In 0 und Funktionsaufruf mvfg_setparam (*MVFGPAR_PHOTO*, PhotoTime,..)) überschreibt PhotoTime den Wert *MVFGPAR_EXPTIME* der dann pro Bild auch geändert werden kann.	

Parameter, die nur mit mvfg_getparam() gelesen werden können:

MVFGPAR_CAPFRAME	Anzahl der tatsächlich gegrabbten Frames bei Verwendung des externen Triggers mit Triggermode 5. (MVFGPAR_EXTPHOTOFLAG = "1", MVFGPAR_TRIGGERMODE = "5") Zweckmäßige Verwendung dieses Parameters mit MVFGPAR_REQFRAME < 0 (d.h. Grabben von mehr als 1 Frame ist angefordert) bei Einsatz von Zeilenkameras.
MVFGPAR_FORMAT_INFO	Struktur FORMAT_INFO mit Informationen über das eingezogene Bild Strukturelemente:

iNumberOfPlanes	Anzahl der Panes, die der Grabber einliest.
iChannelsPerPlane	Anzahl der Kanäle pro Planeepräsentiert z.B. die Farben bei RGB 8:8:8 = 3 Kanäle)
iBitsPerChannel [ ]	Array mit den Anzahlen der Bits pro Kanal. (z.B. bei RGB 5:6:5 = { 5, 6, 5 }, bei 8 bit B&W = { 8 })
iOffsetNIOC	Der Offset zum nächsten Pixel eines Kanals in Bytes. (z.B. bei RGB 5:6:5 (16 bit) = 2 bei RGB 8:8:8 (24 bit) = 3 bei B&W 8 bit = 1 bei B&W 10 bit = 2)
lImageWidth	Die tatsächliche Breite eines darzustellenden Bildes in Pixel.
lImageHeight	Die tatsächliche Höhe eines darzustellenden Bildes in Pixel. (Blacklines sind schon abgezogen.)
lLineSize	Länge einer eingelesenen Zeile in Bytes (Abhängig von der Bildbreite und vom Bildformat). (z.B. Breite = 640 Pixel, RGB 8:8:8 lLineSize = 640 * 3 = 1920 Bytes)
lPlaneSize	Die Größe einer Plane in Byte.
lFrameSize	Die Größe der eingelesenen Daten (alle Planes in Byte.
lColorFormat	MVFG_RGB = Farbbild, MVFG_GRAY = Graustufenbild, MVFG_SPECIAL = besondere Pixelanordung, Details dazu verfügbar über Parmeter MVFGPAR_FORMT_IDENT, siehe nächste Seite

MVFGPAR_FORMAT_IDENT	Kennzeichen spezieller Pixelanordnungen im Grabberspeicher für alle jene Fälle, in denen FORMAT_INFO für lColorFormat den Wert MVFG_SPECIAL besitzt.

Kennzeichen	Bedeutung
PA_BW 8 _2LINES_EO	8 Bit B&W – Inhalt zweier Zeilen als gerade und ungerade Pixel, siehe Kapitel 4.5.4
PA_BW 8 _HALF_EO	8 Bit B&W – erste Hälfte einer Zeile als gerade Pixel, zweite Hälfte einer Zeile als ungerade Pixel, siehe Kapitel 4.5.5
PA_BW 8 _HALF_EO_REVERSE	8 Bit B&W – erste Hälfte einer Zeile als gerade Pixel, zweite Hälfte einer Zeile als ungerade Pixel in umgekehrter Reihenfolge, siehe Kapitel 4.5.6
PA_BW10_HALF_EO	10 Bit B&W – erste Hälfte einer Zeile als gerade Pixel, zweite Hälfte einer Zeile als ungerade Pixel, siehe Kapitel 4.5.10
PA_BW10_HALF_EO_REVERSE	10 Bit B&W – erste Hälfte einer Zeile als gerade Pixel, zweite Hälfte einer Zeile als ungerade Pixel in umgekehrter Reihenfolge, siehe Kapitel 4.5.11

MVFGPAR_DEVICE_INFO_EX	Struktur Z_MVFG_INFO_EX mit Informationen zur installierten Framegrabber Hard- und Software. Die Funktion steht nur ab INSPECTA SetupVersion 4.34 zur Verfügung!

Kennzeichen	Bedeutung
version	Version dieser Struktur (**muss vor Aufruf der Funktion mit 2 initialisiert werden**)
size	Größe dieser Struktur in Bytes (**muss vor Aufruf der Funktion mit der Strukturgröße initialisiert werden**)
driver_version_ms	Device Driver Versionsnummer (Main Version Number)
driver_version_ls	Device Driver Versionsnummer (Sub Version Number)
dll_version_ms	Version der DLL ‚**MVFGD32.DLL**' (Main Version Number)
dll_version_ls	Version der DLL ‚**MVFGD32.DLL**' (Sub Version Number)
hw_type	Framegrabbertyp: 0 = Unknown 1 = Inspecta-1 (MVFG) 2 = Inspecta-2 3 = Inspecta-3 4 = Inspecta-4 5 = Inspecta-5
hw_sub_type	Hardware Interface des Framegrabbers: 0 = Unknown 1 = ISA 2 = PCI 3 = PCI-X 4 = PCI-Express
hw_vendorID	Vendor ID des Framegrabbers
hw_deviceID	Device ID des Framegrabbers: 0x1234 = Inspecta 2/3 0x4d40 - 0x4d43 = Inspecta-4A/-4AE 0x4d44 - 0x4d47 = Inspecta-4D 0x4d48 - 0x4d4b = Inspecta-4C/-4CE 0x4d50 = Inspecta-5 0x4d5d = Inspecta-5DB
hw_revID	Revision des Framegrabbers
hw_snr	Seriennummer des Framegrabbers
hw_imp	Mikrotron internal rev. Number
hw_sub_system_id	Sub-System ID des Framegrabbers

3.3 mvfg_getbufptr(DeviceNumber)

Synopsis:

```
void * WINAPI mvfg_getbufptr( LONG DeviceNumber )
```

Beschreibung:
Die Funktion liefert einen Zeiger auf den Anfang des Bilddaten-Puffers des Inspectas zurück. Dieser Zeiger wird benötigt, um das Bild, das von mvfg_grab geliefert wurde, zu verwenden. Die Funktion darf nur aufgerufen werden, wenn der Treiber aktiv (geöffnet) ist.

Beispiel:
siehe Beispiel Bildeinzug und Maße des Bildes

Parameter:

LONG DeviceNumber
Grabber-Nummer (0 bis 3)

Funktionswert (void *):

Zeiger
auf den Anfang des Bilddaten-Puffers des Inspectas
NULL
Fehler

Die Funktion mvfg_errmessage kann hier nicht angewendet werden, da der Rückgabewert ein Zeiger auf einen Puffer darstellt.

3.4 mvfg_grab(iCommand, DeviceNumber)

Synopsis:

```
LONG WINAPI mvfg_grab( DWORD  iCommand,
                       LONG   DeviceNumber )
```

Beschreibung:

Startet die Belichtung, überprüft den gegenwärtigen Status und tauscht im Double Buffer-Mode (standard) den Speicherblock aus.
Ein Bild wird je nach Einstellungen in den Bilddaten-Puffer eingelesen.

Beispiel:
siehe Beispiel Bildeinzug und Maße des Bildes

Parameter:

DWORD iCommand
gewünschtes Verhalten (siehe Tabelle mit entsprechenden Konstanten auf der nächsten Seite).

LONG DeviceNumber
Grabber-Nummer (0 bis 3)

Funktionswert (LONG):
Abhängig von iCommand (in der untenstehenden Tabelle aufgelistet).

iCommand	**Funktionswerte von mfvg_grab() und deren Bedeutung**	
alle	Funktionswerte, die für alle Parameter iCommand vorkommen können:	
	Funktionswert:	Bedeutung:
	EMVFG_NO_VXD	Fehler: Inspecta nicht installiert
	EMVFG_NOT_OPEN	Fehler: Der Treiber wurde nicht mit mvfg_open geöffnet oder er wurde schon wieder geschlossen.
GRAB_WAIT	zieht ein Bild ein und kehrt erst nach Ende der Aufzeichnung zurück	
	Funktionswert:	Bedeutung:
	MVFG_GRAB_READY	Bildeinzug erfolgreich
	EMVFG_TIMEOUT	Fehler: Bild konnte nicht innerhalb des Timeouts eingezogen werden
GRAB_NOWAIT	stößt einen Bildeinzug an, ohne auf die Beendigung der Aufzeichnung zu warten	
	Funktionswert:	Bedeutung:
	MVFG_OK	Bildeinzug wurde angestoßen
GET_STATUS	Statusabfrage nach Anstoß des Bildeinzuges	
	Funktionswert:	Bedeutung:
	MVFG_GRAB_READY	Bildeinzug erfolgreich abgeschlossen
	MVFG_NOT_READY	Bildeinzug noch nicht fertig
	EMVFG_TIMEOUT	Fehler: Bild konnte nicht innerhalb des Timeouts eingezogen werden
GET_STATUS_WAIT	wartet auf die Fertigstellung eines Bildeinzuges	
	Funktionswert:	Bedeutung:
	MVFG_GRAB_READY	Bildeinzug erfolgreich
	EMVFG_TIMEOUT	Fehler: Bild nicht innerhalb des Timeouts eingezogen
GRAB_STOP	Stoppt einen ggf. laufenden Bildeinzug mit warten auf externen Trigger	
	Funktionswert:	Bedeutung:
	MVFG_OK	Warten auf Trigger und ggf. Bildeinzug abgebrochen

3.5 mvfg_close(DeviceNumber)

Synopsis:

```
LONG WINAPI mvfg_close( LONG DeviceNumber )
```

Beschreibung:
Diese Funktion stoppt den Treiber und deaktiviert ihn.

Beispiel:
siehe Beispiel Öffnen und Schließen des Treibers

Parameter:

LONG DeviceNumber
Grabber-Nummer (0 bis 3)

Funktionswert (LONG)

MVFG_OK
Deaktivierung erfolgreich

EMVFG_NO_VXD
Fehler: Der Grabber ist nicht installiert

3.6 mvfg_errmessage(iCode)

Synopsis:

```
LONG WINAPI mvfg_errmessage( LONG iCode )
```

Beschreibung:
Die Funktion interpretiert den Rückgabewert einer anderen MVFG-Funktion (die Fehlercodes zurückgeben kann) und gibt, falls erforderlich, eine Klartext-Fehlermeldung zum MVFG-Fehler aus. Falls MVFG_OK übergeben wird, passiert nichts. Der zuvor übergebene Code wird zur weiteren Verarbeitung unverändert wieder als Rückgabewert zurückgegeben.

Beispiel:
siehe Beispiel Öffnen und Schließen des Treibers

Parameter:

LONG iCode

Rückgabewert von einer MVFG-Funktion (Fehlercode oder MVFG_OK).
Der Wert darf nicht von mvfg_getbufptr stammen, da diese Funktion keinen Fehlercode, sondern einen Zeiger zurückliefert.
Alle anderen Level1-Funktionen liefern einen Fehlercode zurück.

Funktionswert (LONG):

Der Übergabeparameter iCode wird als Funktionswert wieder zurückgeliefert.

3.6.1 Beispiele

Die Beispiele müssen in ein Windows-Projekt eingebaut werden, damit sie funktionieren. Dazu muss die Datei **mvfgd32.lib** mit dem Linker eingebunden werden.

Für eigene Programme muss auch immer die Header-Datei **„mvfgdrv.h“** verwendet werden.

3.7 Öffnen und Schließen des Treibers

```
/*  This example opens the driver and sets the
 *  parameters for the Testmode. The TestMode is
 *  defined in the section "TestMode” in the file
 *  "Inspecta-XX.CAM”. The first grabber (number 0)
 *  from up to four grabbers (under WinNT) is used
 *  here (Usually only one grabber is installed.) If
 *  the camera-profile-file could not be found you
 *  have to put the complete path like
 *  "C:\\Inspecta-4A.CAM” to the mvfg_open-parameter.
 */

#include "windows.h"
#include "mvfgdrv.h"      //  for the mvfgd32.dll

//  Windows main-function
int APIENTRY WinMain ( HINSTANCE hInstance,
                       HINSTANCE hPrevInstance,
                       LPSTR     lpCmdLine,
                       int       nCmdShow )
{
    LONG iRc;      //  return-value of this function

    //  Opens the driver, loads and sets the profile.
    //  You can also use another profile than
    //  "TestMode” to use your camera.
    iRc  =  mvfg_open( "Inspecta-3.cam;Testmode", 0 );

    //  If mvfg_open returned another value than
    //  MVFG_OK a message is shown.
    mvfg_errmessage( iRc );

    if ( iRc == MVFG_OK )
    {
        //  Puts a message to the screen that the
        //  driver has been opened.
        MessageBox( NULL,   "Driver opened.”,
                    "MVFG”, MB_OK );
    }

    //  If the driver has been opened, it must be
    //  closed by mvfg_close at the end of the program
    iRc  =  mvfg_close( 0 );
    mvfg_errmessage( iRc );

    if ( iRc == MVFG_OK )
    {
        //  Puts a message to the screen.
        MessageBox( NULL,   "Driver closed.”,
                    "MVFG”, MB_OK );
    }

    //  If an error occurred during mvfg_close, the
    //  error-code is returned.
    //  If the driver was not open, nothing happens.
    return iRc;
}
```

3.8 Setzen und Lesen von Parametern

```
/*  This example sets the white-level new.
 *  You can use a camera-mode for your camera.
 */

#include "windows.h"
#include "mvfgdrv.h"      //  for the mvfgd32.dll

//  Windows main-function
int APIENTRY WinMain( HINSTANCE  hInstance,
                      HINSTANCE  hPrevInstance,
                      LPSTR      lpCmdLine,
                      int        nCmdShow )
{
    char acBuffer[256]; //  buffer for the messages
    LONG dwValue;       //  buffer for the white-level

    //  Open the driver, load and set the profile.
    //  You can use another profile than “TestMode”
    mvfg_errmessage( mvfg_open("Inspecta-3.cam;Testmode", 0));

    //  Read the actual white-level-value and put a
    //  message with it to the screen.
    //  The value for the white-level is written to
    //  dwValue. The instance dwValue must have the
    //  right type. (here the type must be LONG).
    _mvfg_getparam_ (MVFGPAR_WHITELEVEL, &dwValue, 0 );

    //  Windows-funktions to shwo a message
    wsprintf(  acBuffer,
               “White-level before setting: %d”,
                dwValue );
    MessageBox( NULL, acBuffer, “MVFG”, MB_OK );

    //  Now the white-level is set to 45.
    mvfg_errmessage( mvfg_setparam( MVFGPAR_WHITELEVEL, “45”, 0 ));

    //  Read the new white-level-value and put a
    //  message to the screen again.
    _mvfg_getparam_( MVFGPAR_WHITELEVEL, &dwValue, 0 );

    //  Put a message to the screen.
    wsprintf( acBuffer,
              “White-level now: %d”,
              dwValue );
    MessageBox( NULL, acBuffer, “MVFG”, MB_OK );

    //  Close the driver.
    mvfg_close( 0 );

    return 0;
}
```

3.9 Bildeinzug und Maße des Bildes

```
/*  This example shows how to get the dimensions of
 *  an image. For this the structure FORMAT_INFO is
 *  used. This structure is defined in the file
 *  MVFGDRV.H.
 *  You can use this functions in your own program.
 *  You must use mvfgd32.lib for the mvfgd32.dll.
 */

//  used the first grabber
#define    GRAB_ID    0

//  include the mvfg
#include  "mvfgdrv.h"

//  decleration of global variables
LONG lFrameWidth;
LONG lFrameHeight;
LONG lFrameSize;

//  function declerations
LONG InitMvfg( void );
void GetDimensions( void );
LONG GrabAndCopyImage( void* pBitMap );


/*  Call this function from your program to initialize
 *  he frame-grabber and the camera(s)
 */
LONG InitMvfg()
{
    int iRc;

    //  initialize the grabber and the camera
    iRc = mvfg_open("Inspecta-3.cam;Testmode",GRAB_ID)

    if( iRc  !=  MVFG_OK )
        return mvfg_errmessage( iRc );
    else
        GetDimensions();

    return MVFG_OK;
}


/*  Call this function every time a camera-parameter
 *  changes.
 */
void GetDimensions( void )
{
    FORMAT_INFO sFormat;

    //  The buffer (second parameter) must be of the
    //  right type. Here the type must be FORMAT_INFO.
    _mvfg_getparam_ (MVFGPAR_FORMAT_INFO, &sFormat, GRAB_ID );

    //  get the width of the image in pixel
    lFrameWidth    = sFormat.lImageWidth;

    //  get the hight of the image in pixel / lines
    //  (all planes)
    lFrameHeight   = sFormat.lImageHeight *
```

```
                    sFormat.iNumberOfPlanes;

    //  get the size of the image in byte (here all
    //  planes)
    lFrameSize     = sFormat.lFrameSize;
}


/*  This function you can use to copy a grabbed image
 *  into another memory-area.
 *  Be sure that the buffer pBitmap is large enough
 *  (at least lFrameSize bytes)
 */
LONG GrabAndCopyImage( void* pBitmap )
{
    int iRc;

    iRc = mvfg_grab( GRAB_WAIT, GRAB_ID );
    memcpy( pBitmap, mvfg_getbufptr( GRAB_ID ), FrameSize );

    return mvfg_errmessage( iRc );
}


/*-----------------------------------------------------*/

/*  The main-function could be WinMain or some thread
 *  or so.
 */
"MAINFUNCTION()"
{
    ...  //  more code

    void * pBitmap;

    //  open and initialize the grabber and the camera
    if( InitMvfg()  ==  MVFG_OK )
    {
        //  Allocate memory for the image-copy.
        //  You can also create a Windows-bitmap here
        //  to display the image.
        pBitmap = malloc( lFrameSize );
    }
    else
    {
        return -1;
    }

    ...  //  more code

    /*  Grab the picture and copy it to the allocated
     *  memory. For example you can create a Windows-
     *  bitmap to display the image. If you use this
     *  in a loop you will get a livepicture.
     */
    GrabAndCopyImage( pBitmap );

    ...  //  more code

    mvfg_close( GRAB_ID );
}
```

4 Begriffsdefinitionen

4.1 Black Lines

Zeilen am Anfang bzw. am Ende eines Kamerabildes, die von der Kamera eingelesen werden, aber keine Bildinformationen enthalten. Diese Zeilen werden vom Grabber nicht eingezogen. Die Eingezogene Zeilenzahl ist somit:
Kamera-Zeilenzahl - blacklines (am Anfang und am Ende jedes Bildes)
Bei Zeilenkameras gibt es keine blacklines.

4.2 Frames

Ein Frame ist die Gesamtheit der Bilddaten, die vom Grabber zwischen zwei VSYNC Signalen eingezogen werden. Ein Frame besteht aus einem oder mehreren Planes.

4.3 Inspecta-XX.cam

Datei mit allen vorgefertigten Kameraprofilen, die von Mikrotron für den Inspecta-XX zur Verfügung gestellt werden. XX steht dabei für die Grabber-Bezeichnung.
(Beispiel: Profile von Inspecta-4A in Inspecta-4A.cam)
Die Profile können durch VCAM oder per Hand verändert werden und die Profildatei kann dann neu gespeichert werden.
In der Datei sind folgende Parameter enthalten, die von Hand gesetzt werden können:

Parameter in Datei	Parameter für Funktion mvfg_setparam(...)
CamMode	MVFGPAR_CAMMODE
LineLen	MVFGPAR_LINELEN
NumLin	MVFGPAR_NUMLIN
BlackLinesBegin	MVFGPAR_BLACKLINESBEGIN
BlackLinesEnd	MVFGPAR_BLACKLINESEND
BlankTime	MVFGPAR_BLANKTIME
WhiteLevel	MVFGPAR_WHITELEVEL
BlackLevel	MVFGPAR_BLACKLEVEL
Videonorm	MVFGPAR_VIDEONORM
Interlaced	MVFGPAR_INTERLACED
ReqFrame	MVFGPAR_REQFRAME
CamSel	MVFGPAR_CAMSEL
DigSel	MVFGPAR_DIGSEL
Photo	MVFGPAR_PHOTO
PhotoFlag	MVFGPAR_PHOTOFLAG
ScanPeriod	MVFGPAR_SCANPERIOD
ContinuousFlag	MVFGPAR_CONTINUOUSFLAG
ExtPhotoFlag	MVFGPAR_EXTPHOTOFLAG
TriggerMode	MVFGPAR_TRIGGERMODE
Timeout	MVFGPAR_TIMEOUT

Parameter in Datei	Parameter für Funktion mvfg_setparam(...)
EnaEnc	MVFGPAR_ENAENC
Divider	MVFGPAR_DIVIDER
ScanRate	MVFGPAR_SCANRATE
ExpTime	MVFGPAR_EXPTIME
Weitere Parameter:	
BufferMode	0 = Singlebuffer / 1 = Doublebuffer
CamString	Name (mit Pfad) einer binären Datei, die über die serielle Schnittstelle des Inspecta-4 an die angeschlossene Kamera gesendet wird.

Keine anderen Parameter verändern!

4.4 Planes

Ein Plane ist ein Bestandteil eines Frames, in dem Bilddaten gespeichert werden. Ein Frame besteht aus einem oder mehreren Planes.
Mehrere planes ergeben sich wenn

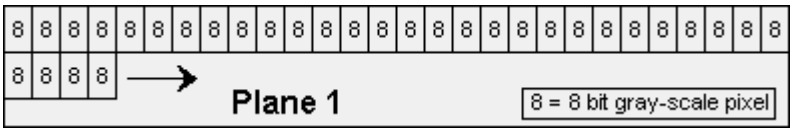
- eine RGB-Kamera über 3 Kanäle 3 planes (rot-plane, grün-plane, blau-plane) erzeugt. (siehe Speicherformate)
- die beiden Halbbilder eine interlaced-Kamera einzeln gespeichert werden. (siehe Speicherformate)
- Bilder von mehreren Kameras gleichzeitig eingezogen werden (je ein oder mehrere planes pro Kamera). (siehe Speicherformate)

4.5 Speicherformate

	**Die Darstellungen der Speicherformate entsprechen im Speicher nicht der Anordnung in Bytes. Die Anordnung im Speicher wird im Text darunter beschrieben.**

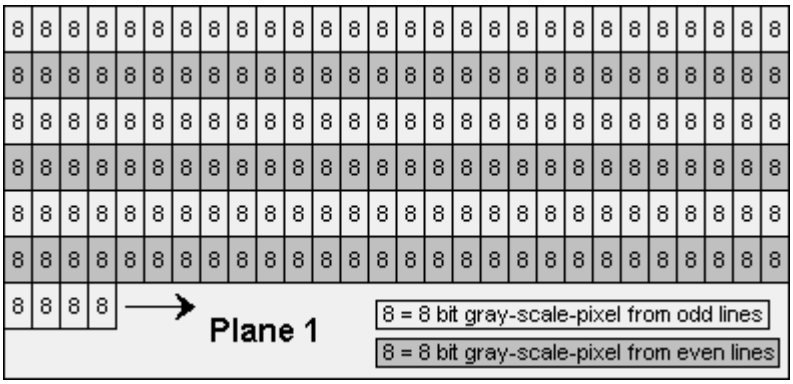
	**Frame Grabber vom Typ Inspecta 4 unterstützen nur gepackte Formate (keine _Planes).**

4.5.1 8 Bit B&W



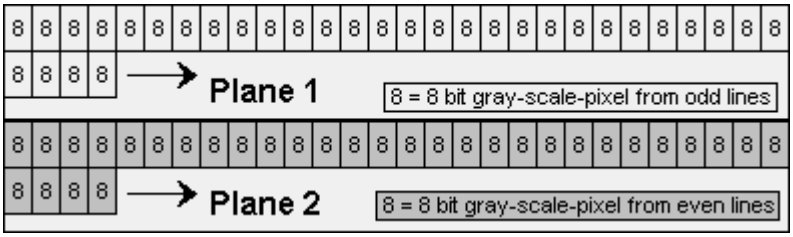
Die Pixel werden als Bytes nacheinander in den Speicher geschrieben.
Gegebenenfalls ist eine Ausrichtung an 4-Byte-Grenzen wie unter „“ beschrieben erforderlich.

4.5.2 8 Bit B&W - interlaced



Zuerst werden alle ungeraden dann alle geraden Zeile in den Speicher eingelesen. Beim schreiben der ungeraden Zeile wird immer eine Zeile freigelassen. Die freien Zeilen werden denn mit den geraden Zeilen aufgefüllt. Danach steht das komplette Bild wie beim Format "8 Bit B&W" im Speicher.
Gegebenenfalls ist eine Ausrichtung an 4-Byte-Grenzen wie unter "24 bit RGB packed" beschrieben erforderlich.

4.5.3 8 Bit B&W - interlaced - 2 Halbbilder



Die beiden Halbbilder der Kamera werden in zwei _Planes geschrieben. Im ersten plane stehen die ungeraden Zeilen, im zweiten die geraden. Um das komplette Bild zu erhalten, müssen die beiden Halbbilder wieder ineinandergekämmt werden.
Gegebenenfalls ist eine Ausrichtung an 4-Byte-Grenzen wie unter "24 bit RGB packed" beschrieben erforderlich.
Um von einer interlaced-Kamera ein komplettes Bild in den Speicher einzulesen, das schon eingekämmt ist, benutzt man das Format "8 Bit B&W - interlaced".

4.5.4 8 Bit B&W – Inhalt zweier Zeilen als gerade und ungerade Pixel

In einer Zeile des Grabberspeichers sind 2 Zeilen einer 8 Bit schwarz-weiß Kamera gepackt: Die geraden Pixel entsprechen einer Zeile, die ungeraden der nächsten Zeile. Folgendes beispielhaftes Abbild der Kamera-Pixel im Grabberspeicher

P00	P10	P01	P11	P03	P13	P04	P14	P05	P15	P06	P16
P20	P30	P21	P31	P23	P33	P24	P34	P25	P35	P26	P36

wird von folgendem Bild einer Kamera geliefert:

P00	P01	P03	P04	P05	P06
P10	P11	P13	P14	P15	P16
P20	P21	P23	P24	P25	P26
P30	P31	P33	P34	P35	P36

P00 = Pixel 0, Zeile 0 des Kamera-Bildes
P01 = Pixel 1, Zeile 0 des Kamera-Bildes
P02 = Pixel 2, Zeile 0 des Kamera-Bildes
.....
P36 = Pixel 6, Zeile 3 des Kamera-Bildes

4.5.5 8 Bit B&W – erste Hälfte einer Zeile als gerade Pixel, zweite Hälfte einer Zeile als ungerade Pixel

In einer Zeile des Grabberspeichers ist eine Zeile einer 8 Bit schwarz-weiß Kamera gepackt: Die geraden Pixel entsprechen der ersten Hälfte einer Zeile, die ungeraden der zweiten Hälfte einer Zeile. Folgendes beispielhaftes Abbild der Kamera-Pixel im Grabberspeicher

P00	P04	P01	P05	P03	P06
P10	P14	P11	P15	P13	P16
P20	P24	P21	P25	P23	P26
P30	P34	P31	P35	P33	P36

wird von folgendem Bild einer Kamera geliefert:

P00	P01	P03	P04	P05	P06
P10	P11	P13	P14	P15	P16
P20	P21	P23	P24	P25	P26
P30	P31	P33	P34	P35	P36

P00 = Pixel 0, Zeile 0 des Kamera-Bildes
P01 = Pixel 1, Zeile 0 des Kamera-Bildes
P02 = Pixel 2, Zeile 0 des Kamera-Bildes
.....
P36 = Pixel 6, Zeile 3 des Kamera-Bildes

4.5.6 8 Bit B&W – erste Hälfte einer Zeile als gerade Pixel, zweite Hälfte einer Zeile als ungerade Pixel in umgekehrter Reihenfolge

In einer Zeile des Grabberspeichers ist eine Zeile einer 8 Bit schwarz-weiß Kamera gepackt: Die geraden Pixel entsprechen der ersten Hälfte einer Zeile, die ungeraden der zweiten Hälfte einer Zeile, wobei diese zweite Hälfte in umgekehrter Reihenfolge abgespeichert wurde. Folgendes beispielhaftes Abbild der Kamera-Pixel im Grabberspeicher

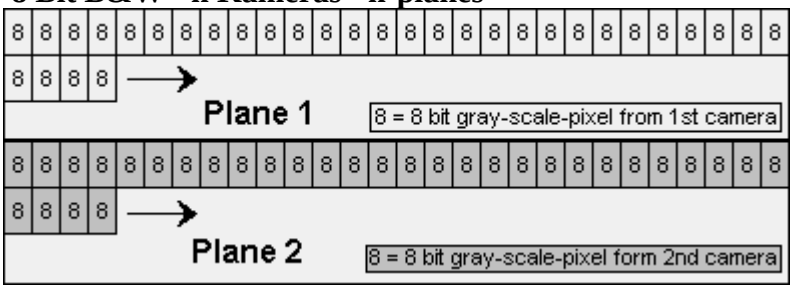
P00	P06	P01	P05	P03	P04
P10	P16	P11	P15	P13	P14
P20	P26	P21	P25	P23	P24
P30	P36	P31	P35	P33	P34

wird von folgendem Bild einer Kamera geliefert:

P00	P01	P03	P04	P05	P06
P10	P11	P13	P14	P15	P16
P20	P21	P23	P24	P25	P26
P30	P31	P33	P34	P35	P36

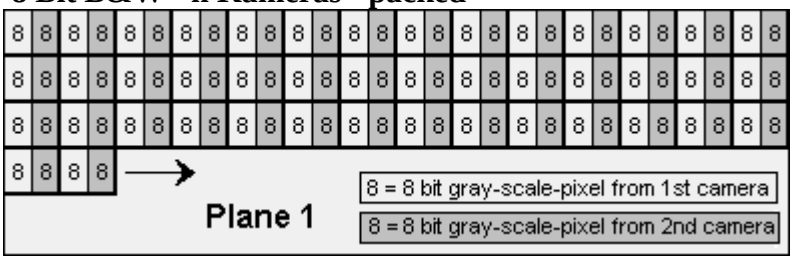
P00 = Pixel 0, Zeile 0 des Kamera-Bildes
P01 = Pixel 1, Zeile 0 des Kamera-Bildes
P02 = Pixel 2, Zeile 0 des Kamera-Bildes
.....
P36 = Pixel 6, Zeile 3 des Kamera-Bildes

4.5.7 8 Bit B&W - n Kameras - n-planes



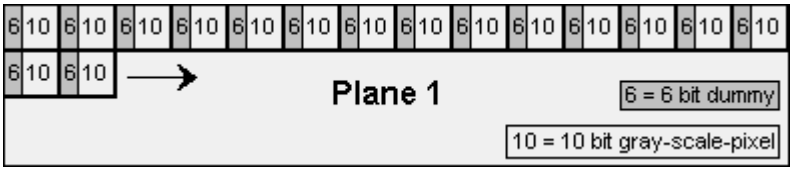
Die Bilder von 2, 3 oder 4 Kameras werden in 2, 3 oder 4 Planes geschrieben (je Kamerabild ein plane).
Gegebenenfalls ist eine Ausrichtung an 4-Byte-Grenzen wie unter "24 bit RGB packed" beschrieben erforderlich.

4.5.8 8 Bit B&W - n Kameras - packed



Die einzelnen Pixel der 2, 3 oder 4 Kameras werden nacheinander (in ein plane) in den Speicher geschrieben.

4.5.9 10 Bit B&W



Es werden immer 2 Byte als Word in den Speicher geschrieben. Die höherwertigen 6 Bit sind Dummy-Bits. Die niederwertigen 10 Bits eines Words werden jeweils als Graustufenwert betrachtet.
Dabei muss beachtet werden, dass ein Word im Speicher anders organisiert ist als die binäre Darstellung einer 16-Bit-Zahl. Die beiden Bytes des Words sind vertauscht, so dass das zweite Byte im Speicher das höherwertige ist.

Die 6 Dummy-Bits werden (je nach Modus) nicht verändert (müssen bei Bedarf auf Null gesetzt werden).

Beispiel:

Binäre Darstellung:

$b_{15}b_{14}b_{13}b_{12}b_{11}b_{10}b_9b_8 \quad b_7b_6b_5b_4b_3b_2b_1b_0$

Darstellung im Speicher:

$b_7b_6b_5b_4b_3b_2b_1b_0 \quad b_{15}b_{14}b_{13}b_{12}b_{11}b_{10}b_9b_8$

Gegebenenfalls ist eine Ausrichtung an 4-Byte-Grenzen wie unter "24 bit RGB packed" beschrieben erforderlich.

4.5.10 10 Bit B&W – erste Hälfte einer Zeile als gerade Pixel, zweite Hälfte einer Zeile als ungerade Pixel

Die Anordnung der Pixel entspricht der Beschreibung unter 4.5.5
„8 Bit B&W – erste Hälfte einer Zeile als gerade Pixel, zweite Hälfte einer Zeile als ungerade Pixel“ auf Seite 35 bis auf die Anzahl Bits pro Pixel (10 statt 8).

4.5.11 10 Bit B&W – erste Hälfte einer Zeile als gerade Pixel, zweite Hälfte einer Zeile als ungerade Pixel in umgekehrter Reihenfolge

Die Anordnung der Pixel entspricht der Beschreibung unter 4.5.6
„8 Bit B&W – erste Hälfte einer Zeile als gerade Pixel, zweite Hälfte einer Zeile als ungerade Pixel in umgekehrter Reihenfolge“ auf Seite 35 bis auf die Anzahl Bits pro Pixel (10 statt 8).

4.5.12 16 Bit RGB

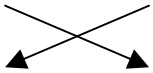
Die Pixel werden als Words nach einander geschrieben.
Dabei muss beachtet werden, dass ein Word im Speicher anders organisiert ist als die binäre Darstellung einer 16-Bit-Zahl. Die beiden Bytes des Words sind vertauscht, so dass das zweite Byte im Speicher das höherwertige ist.

Beispiel:

Binäre Darstellung:

$$r_0r_1r_2r_3r_4g_0g_1g_2 \quad g_3g_4g_5b_0b_1b_2b_3b_4$$

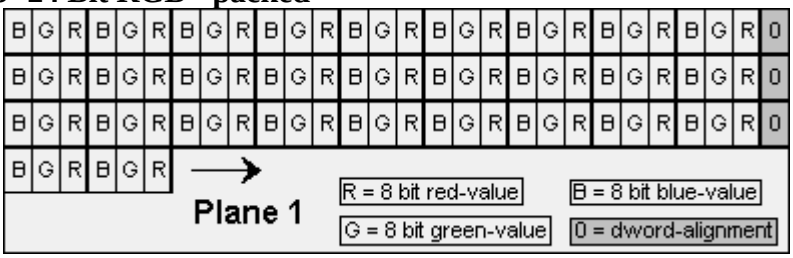
Darstellung im Speicher:



$$g_3g_4g_5b_0b_1b_2b_3b_4\ldots. \quad r_0r_1r_2r_3r_4g_0g_1g_2$$

Gegebenenfalls ist eine Ausrichtung an 4-Byte-Grenzen wie unter "24 bit RGB packed" beschrieben erforderlich.

4.5.13 24 Bit RGB - packed



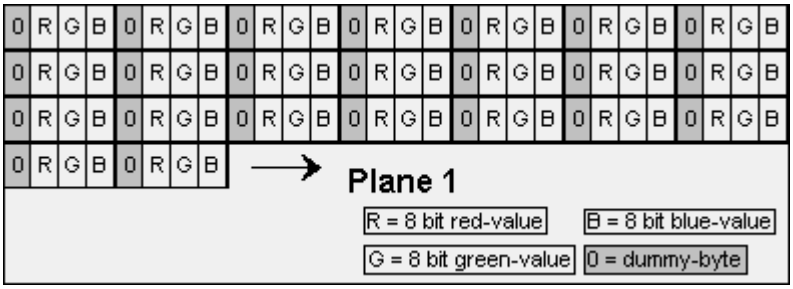
Es werden immer 3 Bytes für die einzelnen Pixel geschrieben. Das Ende einer Zeile muss aufgefüllt werden, bis die Zeile durch vier Bytes teilbar ist. Ist die Länge einer Zeile bereits durch vier Bytes teilbar, so wird nicht aufgefüllt. (Die Zeilenlängen darf immer nur ein Vielfaches von einem Dword sein)
Bei der Verwendung eines Inspecta entstehen nur Zeilenlängen, die durch vier Bytes teilbar sind, also wird nie aufgefüllt.

	**Die 3 Bytes werden im Speicher in der Anordnung BGR (Blau-Grün-Rot), nicht RGB abgelegt.**

4.5.14 24 Bit RGB - 3-planes

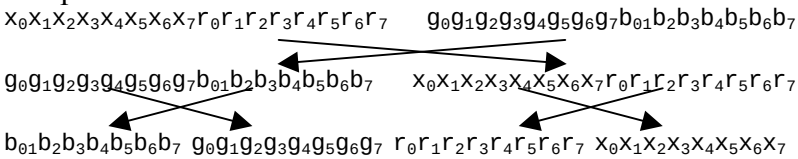
Pro Farbe wird je ein plane mit je einem Byte pro Pixel geschrieben.
Gegebenenfalls ist eine Ausrichtung an 4-Byte-Grenzen wie unter "24 bit RGB packed" beschrieben erforderlich. Die Ausrichtung bezieht sich dann jeweils auf ein plane.

4.5.15 32 Bit xRGB



Die Pixel werden als Dwords geschrieben. Dabei entsehen 3 Daten-Bytes und 1 Füllbyte. Das Füllbyte ist das höherwertige Byte im höherwertigen Word.
Im Speicher wird ein Dword anders organisiert als im obigen Bild zu sehen ist. Die beiden Words des Dwords werden vertauscht und dann werden innerhalb der beiden Words jeweils die beiden Bytes vertauscht. Dies entspricht einer invertierten Reihenfolge in der Anordnung der Bytes innerhalb eines Dwords.

Beispiel:



Es ergibt sich also im Speicher die Reihenfolge
BGRx BGRx BGRx ...

4.6 Testmode

Interner Testmodus des Grabbers, der eine Kamera simuliert.

4.7 VSYNC-Signal

Ein Signal, das von der Kamera zwischen zwei Bildern gesendet wird.