INSPECTA

Der Realzeit Frame Grabber für den PCI-BUS

Software Beschreibung
Rev. 1.49

1 Allgemeines

1.1 Aufbau des Handbuchs

Dieses Handbuch ist für den versierten Anwendungsprogrammierer geschrieben. Es beschreibt die Benutzung der zum INSPECTA gehörenden Software unter Windows und Phar Lap DOS-Extender.

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

Im Anhang ist eine Liste der I/O Ports und eine Beschreibung deren Funktion angegeben. Damit ist eine vollständige Programmierung von INSPECTA-1 auf Hardware-Niveau möglich.

1.2 Revisionsgeschichte

Dieses Handbuch beschreibt die Software:

PCAM.EXE ; ab Apr. 96
MPFGDRV.ASM ; ab Apr. 96

ab Nov. 96 wurden Änderungen für INSPECTA aufgenommen.

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.

1.3 Markenrechte

2 Software

2.1 Installationshinweise für Windows® NT/2000/XP

2.1.1 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).

2.1.2 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.

2.1.3 Registry Einträge

Folgende Schlüssel werden verwendet:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\MpfgNt
```

Sie haben folgende Werte (Beispiel):

```
MemStartPage=0x03000000  ; 48MB Bildspeicher Startadresse
BlockSize=0x01000000     ; 16MB Bildspeicher Länge
DeviceID=0x00001234
MemoryAllocationMode 00000000 ; compatibility mode =0, maxmem mode
= 1
```

```
Devicenumber=0x00000000  ; für ersten INSPECTA, benütze 1,2, und 3
                         ; for bis zu vier INSPECTA
```

Keine andreren Werte ändern!

2.1.4 INSPECTA Installation unter Windows® NT

Es ist die Version: **Workstation 4.0 mit eingespieltem Service Pack mindestens 3** notwendig. Die Installation geschieht 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.

2.1.4.1 Mehrere Inspectas in einem PC unter WinNT

Für vier möglich Inspectas wird 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.

2.1.5 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.

2.1.6 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.

2.1.6.1 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.

2.2 Installationshinweise für Windows® 9x/DOS

2.2.1 INSPECTA Installation ab Windows® 95-SR2

Für die komfortable Installation der INSPECTA Treiber gibt es eine Installations Diskette. Sie kopiert folgende Dateien:

%WINBOOTDIR%\insp2.386
%WINBOOTDIR%\mvfgdrv.dll
%WINBOOTDIR%\mvfgd95.dll
%WINBOOTDIR%\mvfgd32.dll

Das Installations Programm fügt in der SYSTEM.INI Datei in der Sektion [386Enh] den Eintrag „device=INSP2.386“ und eine neue Sektion ein:

```
[INSP2]
BlockSize=400h ; = Länge des zu reservierenden Bildspeichers

[386Enh]
.
.
.
device= INSP2.386
```

Das Schlüsselwort für die Länge des Bildspeichers ist „BlockSize“ und wird in Einheiten von „Pages“ in hexadezimaler Notation eingetragen. Eine Page sind 4096 Bytes.

In bestimmten Situationen und wenn ein größerer Bildspeicher als z.B.B 16MB nötig ist, muß ein anderer Weg gewählt werden.

Die folgende Beschreibung setzt Win95SR2 oder Win98 und eine Treiberversion > 1.65 voraus.

Ab Windows95 SR2 können größere zusammenhängende Speicherblöcke reserviert werden.

Dazu muß in der Sektion **[INSP2]** des SYSTEM.INI das Schlüsselwort XmsStart sowie in der Sektion [386enh] das Schlüsselwort MaxPhysPage eingetragen sein:

Beispiel: Bei 96 MB Hauptspeicher sollen 64 MB als Bildspeicher reserviert werden.

Eintrag in SYSTEM.INI:

```
[INSP2]
BlockSize=4000h
XmsStart=2000h

[386enh]
....
```

MaxPhysPage = 1fffh

Damit ist die Speichergröße: 4000h*4096 = 16384*4096 = 64MB
und beginnt ab MaxPhysPage = 1fffh+1 = XmsStart=2000h = 32MB

Diese Eintragungen müssen manuell ergänzt werden, sie werden **nicht** duch die Setup Diskette durchgeführt.

2.2.2 Inspecta Installation unter DOS Extender

Die DOS-Treiber sind unter www.mikrotron.de abrufbar.

Zur Treiberdiskette für DOSX gehören folgende Module die im Selbstentpacker mpfgxxx!.exe (xxx = Versionsnummer) enthalten sind:

```
MPFGEQU  ASM   Kameraparameter und interne Definitionen
INSP2LIB LIB   Phar-Lap INSPECTA Library
ICAM     EXE   DOS Testprogramm fuer INSPECTA
MVFG     H     DOSX Definitionen
MVFGFNT  H     DOSX Font-Definitionen
MVFGTEXT H     DOSX Text-Definitionen
```

Für den Betrieb des INSPECTA unter Phar Lap DOS Extender muß in der Datei: CONFIG.SYS der Eintrag HIMEM.SYS wie folgt lauten:

DEVICE=C:\DOS\HIMEM.SYS /INT15=XXXX
Der Wert XXXX = Gesamtspeicher (ohne INSPECTA Speicher) in Kbytes - 1088 - [länge von SMARTDRV in KB] - [länge von RAMDRIVE in KB]

Wenn zusätzlich auch SMARTDRV verwendet wird, muß in TNT, 386ASM, 386LINK, und HCD3861.EXE, HCD3862.EXE der Switch -extlow 110000h konfiguriert werden. Zum konfigurieren wird das Phar Lap Programm CFIG386 verwendet. Siehe dazu auch die DOS-Extender Dokumentation.

Der Phar Lap DOS-Extender benutzt den Real-Speicher bis A0000h und den extended Speicher oberhalb der durch -extlow gegebenen niedrigsten Adresse ab der durch den Eintrag /INT15=XXXX+1088 gegebenen höchsten Adresse abwärts.

Ab dieser Adresse reserviert die Funktion mfgtest () eine mit mfg_physlen () konfigurierbare Menge an Speicher für die ausschließliche Nutzung als Bildspeicher.

Der Bildspeicher wird vom Treiber in zwei Hälften aufgeteilt, in die abwechselnd ein Bild geschrieben wird. Die Hälfte wird mit der Funktion mvfg_xchg (), dem Rücksetzen des _fgtv_valid Flags oder mit dem Aufruf der Funktion mvfg_input (timeout) gewechselt.

Die Variablen mfg_lin oder MVFGD_MFG_LIN erhält dadurch einen neuer Wert. Damit muß diese Varible unmittelbar als Pointer auf den Bildspeicher verwendet werden, oder deren Wert nach jedem Speicherblocktausch neu gelesen werden.

Von der Verwendung von EMM386.EXE im CONFIG.SYS wird wegen der dann verlängerten Ladezeit bei großem Hauptspeicher abgeraten.

2.3 Level1 Funktionen

Mit dem Level1 API sind nur wenige Funktionsaufrufe nötig um den INSPECTA zu initialisieren, eine bestimmte Kamera auszuwählen und ein Bild einzuziehen. Level1 Funktionen sind nur in WinNT/2K/XP implementiert. Die Beschreibung dieser Funktionen und Beispiele dazu sind in dem separaten Manual „level1d.pdf“ zu finden, das Sie sich ggf. von unserer homepage www.mikrotron.de herunterladen können.

2.4 Level0 Treiberfunktionen

Die Funktionsaufrufe unter DOS-Extender bzw. Windows unterscheiden sich in der Regel nur durch den Namen. Die Windows-Aufrufe beginnen einheitlich mit **mvfg_**funktionsname. Die DOS-Extender Aufrufe beginnen in der Regel mit **mfg_**funktionsname. Die Beschreibung ist zusammengefaßt. Einige Funktionen gibt es nur unter DOS-Extender (z.B.: alle Bilddarstellungsfunktionen), andere nur unter Windows.

Die im MPFGDRV enhaltenen Funktionen behandeln in der Regel jeweils nur einen einzelnen Aspekt zur Steuerung des INSPECTA. Um eine bestimmte Leistung zu erhalten, müssen öfters mehrere Funktionen hintereinander aufgerufen werden.

2.4.1 Initialisierung

Die Initialisierungsfunktionen bestimmen die Einbindung der INSPECTA Hardware in das Motherboard. Es müssen zum Anfang des Programms folgende Funktionen einmalig aufgerufen werden:

```
mfgtest();              /*  testet den INSPECTA und belegt interne variablen
und I/O
                        Ports */
vmfg_isr(0x10);         /* Stellt den vom PCI-BIOS zugeteilten Interrupt Vektor
                        auf die in MPFGDRVA enthaltene Interrupt Service Routi-
ne */
mfg_int(0x410);         /* Interrupt Quelle ist VD fallende Flanke und wird
                        aktiviert */
```

Von jetzt an können schon Bilder gespeichert werden, wenn diese dargestellt werden sollen, muß auch die Bildausgabehardware initialisiert werden:

```
init928(0);             /* VGA Auflösung 640 * 480 * 8 bpp */
palette928(0);          /* 8 Bit Graupalette mit 4 Overlay Farben */
mfg_clip(0);            /* schaltet ein evtl undefiniertes clip-window aus */
```

Danach müssen Funktionen zur Bildgeberauswahl und Speicherverwaltung aufgerufen werden. Beispiele sind in den folgenden Abschnitten angegeben.

Vor Beendigung der Applikation muß noch der Interrupt wieder ausgeschaltet werden.

```
mfg_int(0);       /* Deselektiert IRQ xx */
vmfg_isr(0);      /* restauriert die ursprünglichen Interrupt Vektoren */
```

Entsprechend diesem Ablauf ist die folgende Beschreibung der Funktionen nach den Kategorien:

Initialisierung, Bildgeberauswahl, Speicherverwaltung, Hilfsmittel und Historie gegliedert.

2.4.1.1 mvfg_test (), mfgtest()

Synopsis: LONG mvfg_test (void);
Beschreibung: Die Funktion übernimmt die Grundinitialisierung und den Test der INSPECTA Hard- und Software.

Returnwerte: == 0 Initialisierung und Test OK !
!= 0 Fehlercode:

-2 memory base address not found ; DOSX only
-3 cant map phys -> lin ; DOSX only
-6 memory banks do not exchange
-8 no clock from self test
-10 no frame data valid from self test
-11 no line data valid from self test
-13 WRONG CPU (386) ; DOSX only

Beispiel:

```
LONG        lMvfgError;

if (lMvfgError = mvfg_test())
{
wsprintf( acPuffer, "Fehler %lh bei Initialisierung des INSPECTA's",
lMvfgError );
...
}
```

2.4.1.2 mvfg_isr (irqmode)

Synopsis: void mvfg_isr (DWORD irqmode);
Beschreibung: Die Funktion legt die Interruptquelle und den Status des verwendeten Hardwareinterrupts des INSPECTA fest (ON/OFF). zusätzlich übernimmt sie die Installation der INSPECTA Interruptroutine.

Der Aufbau des Parameters 'irqmode' ist wie folgt:

Bits 0 - 7 ==0 : Interrupt aus
>0 : Interrupt ein
Bits 8 - 15 Interruptquelle.
Mögliche Werte für Interruptquelle sind im nächsten Abschnitt beschrieben.

Returnwerte: --

Beispiel:

```
/* Für den Betrieb des INSPECTA wird als Interruptquelle der vert. Syn-
chronisationsimpuls (Bit 10 der Maske ) des INSPECTA verwendet. und der
Interrupt eingeschaltet */
mvfg_isr ( 0x0400 | 0x10 );
```

Bemerkung: Diese Funktion gibt es nur unter Windows

2.4.1.3 vmfg_isr (int_mode)

Beschreibung: Passend zum gegebenen IRQx wird ein protected mode Interrupt Vector, der auf die in MPFGDRV enthaltene Service Routine zeigt (isr), geschrieben. Der gewählte IRQx wird aktiviert. Wenn int_mode den Wert 0 hat, dann wird der gewählte IRQx deaktiviert und der vorherige Vector wiederhergestellt..

Bemerkung: Diese Funktion gibt es nur unter DOSX

2.4.1.4 mfg_int (int_mode)

Beschreibung: mfg_int wählt die Interrupt Quelle (z.B.: VD fallende Flanke) und aktiviert oder deaktiviert den vom PCI-BIOS beim Booten des Systems vegebenen IRQ. Eine Interrupt Service Routine muß vorher installiert worden sein. Siehe vmfg_isr (source/irq).

Parameter: ‘int_mode’ kann folgende Werte haben:

Bits:	0..7	= 0:	deactivate INTA
		> 0:	activate INTA
	8..9	= 0	
	10	= ISRC0 interrupt source 0	
	11	= ISRC1 interrupt source 1	
	12..32	= 0	

int_mode = 0x010 ; activated by pixel clock/131072
0x410 ; activated by falling edge of VSYNC
0x810 ; activated by bit 0 of opto input rising edge
0xC10 ; activated by falling edge of HSYNC

Bemerkung: Diese Funktion gibt es nur unter DOSX. Für Win 95/NT siehe mvfg_IntSource ().
Parameter int_mode Beschreibung für INSPECTA, Hardware Revision >=4, IMP Nr.: >= IMP385, Software Rev. >=1.80

2.4.1.5 mvfg_maskint (onoff, mask), mfg_maskint

Synopsis: void mvfg_maskint (onoff, mask);

Beschreibung: Die Funktion mfg_maskint verhindert die angegebenen System Interrupts, oder läßt sie wieder zu. Dies ist notwendig, um z.B.: während der Belichtungssteuerung einer Shutter-Kamera mit variabler Belichtungszeit diese Zeit konstant halten zu können.

Das Applikationsprogramm kann entweder selbst entscheiden, wann eine Aussperrung wirksam sein soll, oder es kann diese Entscheidung dem Treiber überlassen.

Im ersteren Fall ruft das Anwendungsprogramm zu Beginn der kritischen Programmzeilen mvfg_intmask (0, mask), und am Ende des Bereichs mvfg_intmask (1, mask) auf.

Wenn der Treiber entscheiden soll, wird im Applikationsprogramm nur einmal mvfg_intmask (1, mask) aufgerufen.

Parameter: onoff = 0: gewählter Interrupt ausgesperrt.
= 1: gewählter Interrupt wird zugelassen, außer zu bestimmten vom Treiber bestimmten Zeiten.
mask: Auswahl der zu beeinflussenden Interrups:

‘mask’ Bits o..31:

```
;31-16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0
;  0   !  !  !  !  !  !  ! ! ! ! ! ! ! ! ! IRQ0: Timer
;  0   !  !  !  !  !  !  ! ! ! ! ! ! ! ! !-- IRQ1: Keyborad
;  0   !  !  !  !  !  !  ! ! ! ! ! ! ! !---- IRQ2: Interrupt Controller 1
;  0   !  !  !  !  !  !  ! ! ! ! ! ! !------ IRQ3: Serial Port 2
;  0   !  !  !  !  !  !  ! ! ! ! ! !-------- IRQ4: Serial Port 1
;  0   !  !  !  !  !  !  ! ! ! ! !---------- IRQ5: Parallel Port 2
;  0   !  !  !  !  !  !  ! ! ! !------------ IRQ6: Diskette Controller
;  0   !  !  !  !  !  !  ! ! !-------------- IRQ7: Parallel Port 1
;  0   !  !  !  !  !  !  ! !---------------- IRQ8: Real Time Clock
;  0   !  !  !  !  !  !  !------------------ IRQ9: Software redirected to INT0Ah
;  0   !  !  !  !  !  !--------------------- IRQ10: Free
;  0   !  !  !  !  !------------------------ IRQ11: Free
;  0   !  !  !  !-------------------------- IRQ12: Free
;  0   !  !  !---------------------------- IRQ13: Coprozessor
;  0   !  !------------------------------- IRQ14: Fixed Disk Controller
;  0   !---------------------------------- IRQ15: Free
```

2.4.1.6 mvfg_datpnt (), mfg_datpnt ()

Synopsis: LPLONG mvfg_datpnt (void);

Beschreibung: Die Funktion gibt einen Zeiger auf den INSPECTA Arbeitsdatenbereich zurück.

Auf die einzelnen Variablen kann in Windows über die in der Headerdatei MVFGDRV.H' definierten Indizes zugegriffen werden. Unter DOSX werden die Variablen in der Regel direkt benützt. Sie sind daher alle „public“.

Bemerkung: Die Struktur des Datenbereichs und die Bedeutung der Variablen kann im Source-Code von MPFGDRV.ASM nachgelesen werden. Die darin angegebenen Voreinstellungen sind solange gültig, bis sie von entsprechenden Funktionen mit deren Aufrufparametern überschrieben werden.

Returnwerte: --

Beispiel:

```
LPLONG      lpMvfgDat;

lpMvfgDat = mvfg_datpnt ();

/* Anzahl von Pixel / Zeile des akt. definierten Frames ermitteln. */
wsprintf( acPuffer, "Anzahl Pixel / Zeile = %ld",
lpMvfgDat[ MVFGD_LINELEN] );
```

2.4.1.7 mvfg_alloc(void);

Synopsis: HGLOBAL mvfg_alloc(void);

Beschreibung: Die Funktion ermittelt einen globalen Speicherblock Handle auf den INSPECTA - Speicherbereich.

Returnwerte:
!= 0 Handle des Speicherblocks.
== 0 Handle auf Speicherblock konnte nicht ermittelt werden.

Beispiel:

```
HGLOBAL     MvfgHandle;

if ( !(MvfgHandle = mvfg_alloc()))
{
exit ( 1 );
}
```

Bemerkung: Diese Funktion steht nur unter Windows zur Verfügung

2.4.1.8 mvfg_lock(HGLOBAL sel);

Synopsis: LPVOID mvfg_lock(HGLOBAL sel);

Beschreibung: Die Funktion setzt einen Lock auf den INSPECTA Speicherbereich und übergibt dem Aufrufer einen Zeiger auf diesen Bereich.

Returnwerte: != 0 Zeiger auf Speicherbereich.
== 0 Zeiger konnte nicht ermittelt werden.

Beispiel:

```
LPVOID lpMvfgSpeicher;

if (!( lpMvfgSpeicher = mvfg_lock( MvfgHandle )))
{
exit ( 1 );
}
```

Bemerkung: Diese Funktion steht nur unter Windows zur Verfügung

---

2.4.1.9 mvfg_unlock(HGLOBAL sel);

Synopsis: BOOL mvfg_unlock(HGLOBAL sel);
Beschreibung: Gibt den Lock auf den INSPECTA Speicher frei.

Returnwerte: == 0 Lock freigegeben.
!= 0 Lock konnte nicht freigegeben werden.

Beispiel:

```
mvfg_unlock ( MvfgHandle );
```

Bemerkung: Diese Funktion steht nur unter Windows zur Verfügung

---

2.4.1.10 mvfg_size (HGLOBAL sel);

Synopsis: DWORD mvfg_size (HGLOBAL sel);
Beschreibung: Die Funktion gibt die Größe des in den Hauptspeicher eingeblendeten INSPECTA Speicherblocks in Bytes wieder.

Returnwerte: Blockgröße in Bytes.

Beispiel: --

Bemerkung: Diese Funktion steht nur unter Windows zur Verfügung

2.4.1.11 mvfg_contWriteInit (DWORD PhysAddr);

Synopsis: mvfg_contWriteInit (DWORD PhysAddr);

Beschreibung: Die Funktion intialisiert INSPECTA für kontinuierliche Aufzeichnen von langen Bildsequenzen bis maximal ca. 4 Gbyte. DWORD PhysAddr gibt die Anfangsadresse des Bildspeichers an. Wird zusammen mit mvfg_contWrite (NrOfFrames, Circular) verwendet.

Returnwerte: --

Beispiel: --

Bemerkung: Treiberversion > 1.65 und INSPECTA Hardware Rev. >4, IMP Nr. >317 notwendig. Multiplane Kamera Modi sind nicht möglich. Alle Videozeilen außerhalb des vertikalen Sync-Impulses werden aufgezeichnet. Halbbilder werden nicht eingekämmt. Mit mvfg_blank (blanktime) können die schwarzen Pixel zu Beginn der Zeile unterdrückt werden.

2.4.2 Bildgeberauswahl

Die Funktionen zur Bildgeberauswahl stellen die Hardware für die Video-Quelle (Testbild, analog oder digital), bei analog Video deren weiß-, schwarz- oder sync-Schwelle wie deren Kamera Eingang, und das Bildformat (linelen, numlin, blank, blacklines, interlace , mfg_pal) ein.
Als Beispiel wird eine Panasonic WV CD-50 mit internem Takt und Sync (beide Signale von M314 erzeugt) ausgewählt.

```
mfg_modcam(0x28);         /* analoge Kamera ohne Pixel Clock aber mit Syc
                             Eingang  */
mfg_camsel(0);            /* wähle oberen D-Stecker, Kanal rot, als Eingang */

/* die folgenden 7 aufrufe werden nur dann benötigt, wenn die mit mfg_modcam ()
vorgenommene Einstellung dieser Parameter nicht zur Kamera passt. */

mfg_whitelevel(192);      /* Weiß-Schwelle ist  192 * 1,2V/256 = 0,9V */
mfg_blacklevel(32); /* Schwarz-Schwelle ist   32 * 1,2V/256 = 0,15V */
mfg_synclevel(1);         /* Sync-Schwelle ist 125 mV über sync tip,
                             nur wichtig  bei composite video modes 0x78 */
mfg_blank (84);           /* blank Zeit von HD fallender Flanke bis
                             Zeilenbeginn = 2*84 * Pclk = 11,84 usec */
mfg_black(26);            /* Anzahl schwarzer Zeilen vor Bildbeginn die nicht
                             gespeichert  werden sollen = 26 */
h_start(603,472,1,0);     /* 603 == linelen, 472 == numlin: incl der nicht
                             zu speichernden 26 schwarzen Zeilen, 1 == interlace,
                             0 == req_frm : nur Bild 0 wird im Speicher
                             geschrieben */
mfg_pal(1);               /* 625 Zeilen / 50 Hz Format */
```

2.4.2.1 mvfg_modcam (mode), mfg_modcam (mode);

Synopsis: LONG mvfg_modcam (DWORD mode);

Beschreibung: Definition des Kamera-Modus in dem der INSPECTA betrieben wird. Der Modus ist vom Typ der verwendeten Kamera abhängig.

Parameter: Die im Anhang B angegebene Tablelle zeigt die möglichen Werte sortiert nach Wert und Funktion.

Returnwerte: == 0 OK
-1 Unbekannter Modus.
-2 Eine Kamera für externen Clock wurde ausgewählt, der externe Clock wurde aber nicht gefunden.

Beispiel:

```
LONG        lMvfgError;

if  ( lMvfgError = mvfg_modcam ( 0x8C ))   /* PULNIX 9700, digital */
{
wsprintf( acPuffer, "Undefinierter Kameramodus");
...
}
```

mfg_modcam stellt INSPECTA auf die angeschlossene Kamera ein. Die bits 4..30 der mode nr. wählen analog oder digital-video, externen oder internen clock und sync aus. Das low nibble setzt die Kamera Control bits mc0..mc3.

Die globalen Variablen dwords linelen, numlin, interlace, und req_frm werden benutzt, um die Anzahl pixel/Zeile, Anzahl Zeilen pro Vollbild, Interlace Modus, und Bildauswahl zu treffen. Dazu wird die Funktion h_start aufgerufen.

Die globale Variable dword blank verlängert den HD Impuls intern, um die horizontale Schwarzschulter auszublenden. Sie wird durch die Funktion mfg_blank geschrieben und die Verlängerung ausgeführt.

Die schwarz und weiß Schwelle wird durch die globalen Variablen dwords mfg_blacklevel und mfg_whitelevel eingestellt.

Synclevel wird je nach modus auf auf 25mV oder 125mV eingestellt. Dazu wird die Funktion mfg_synclevel aufgerufen.

Die Variable DWORD seq_color definiert die Organisation des Bildspeichers entsprechend der folgenden Tabelle:

```
seq_color       ; 0: eine Plane
                ; 1: three aufeinanderfolgende Planes, offset ist die Variable [pel_frm]
                ; 2: zwei frames interlaced und gleichzeitig (Sony xc7500, Kodak ES 1.0)
                ; 3: zwei Planes non-interlaced
```

Die Variable DWORD colour_type definiert die Pixel Anordnung im Bildspeicher entsprechend folgender Tabelle:

```
colour_type     ; 1 = 3:3:2 8bpp, 1-plane, rgb
                ; 2 = 5:6:5 16bpp, 1-plane, rgb
                ; 3 = 8:8:8 8bpp, 3-planes, rgb
                ; 4 = 8:8:8:8 32bpp, 1-plane, xrgb
                ; 5 = 8:8:8 24bpp, 1-plane, rgb
                ; 6 = 16:16:16 48bpp, 1-plane, rgb
                ; 7 = 8:8:8:8 32bpp, 1-plane, 4-cameras b&w
                ; 8 = 8:8      16bpp, 1-plane, 2-cameras b&w
```

2.4.2.2 mvfg_camsel (camnr), mfg_camsel (camnr)

Synopsis: void mvfg_camsel (DWORD sel);

Beschreibung: 'mvfg_camsel()' wählt eine von sechs (acht mit Zusatz A/D Karte MAD 1020) Kameras als Video Quelle und ggf. auch als Clock und Sync Quelle oder Ziel aus.

camnr	Video	Stecker
0	grün	P1
1	blau	P1
2	rot	P1
3	grün	P1 (MAD1020)
4	grün	P2
5	blau	P2
6	rot	P2
7	grün	P2 (MAD1020)

Bei Verwendung der High-Speed A/D MAD 1020 zusammen mit INSPECTA wird die bisherige Doppelbelegung der camnr 3 und 7 aufgehoben und stattdessen die beiden Eingänge auf der Zusatzkarte MAD 1020 verwendet.

Bei den Kameramodi für mehrkanalige Kameras, (z.B.: Farbkameramodi 0x78, 0xC0, 0xEC, 0x148) wird für Werte 0..3 der Stecker P1 und für Werte 4..7 der Stecker P2 gewählt.

Returnwerte: --

Beispiel: --

mfg_camsel schreibt die globale Variable mfg_camnr.

2.4.2.3 mvfg_digsel (camnr), mfg_digsel (camnr)

Beschreibung: mit mvfg_digsel wird eine über den MUX 5000 Digital Multiplexer angeschlossene Digitalkamera selektiert, wobei dessen Steuerung über den LPT-Anschluß erfolgt. Damit können bei Verwendung von bis zu 4 Multiplexern bis zu 16 Kameras ausgewählt werden. Einzelheiten dazu sind dem Dokument „MUX 5000 Digital Multiplexer, Beschreibung“ zu entnehmen

Returnwerte: --

Beispiel: --

2.4.2.4 mvfg_whitelevel (whitelevel), mfg_whitelevel (whitelevel)

Synopsis: void mvfg_whitelevel (DWORD level);

Beschreibung: Definition der Weißschwelle des Video A/D Wandlers.
Mögliche Werte: 00h - FFh (0 .. 1200 mV).

Returnwerte: --

2.4.2.5 mvfg_blacklevel (blacklevel), mfg_blacklevel (blacklevel)

Synopsis: void mvfg_blacklevel (DWORD level);

Beschreibung: Definition der Schwarzschwelle des Video A/D Wandlers.
Mögliche Werte: 00h - FFh (0 .. 1200 mV).

Returnwerte: --

mfg_blacklevel stellt die untere Schwelle des Video A/D in 256 Stufen zwischen 0 ... 1200 mV ein.
Das Video Signal wird mit seiner vorderen Schwarzschulter auf 0 Volt geklemmt.

2.4.2.6 mvfg_synclevel (synclevel), mfg_synclevel (synclevel)

Synopsis: void mvfg_synclevel (DWORD level);
Beschreibung: Über die Funktion wird der Schwellwert für die Separation der Synchronisationssignale festgelegt.

Mögliche Werte: 0 25 mV
1 125 mV

Returnwerte: --

Beispiel: --

2.4.2.7 mvfg_blank (blank_time) , mfg_blank (blank_time)

Synopsis: void mvfg_blank (DWORD blank);

Beschreibung: Die Funktion definiert die Anzahl von Pixelclocks * 2 zwischen der fallenden Flanke des horizontalen Synchronisationssignals bis zum 1. gültigen Pixel der Zeile.

Dieser Wert ist vom Typ der Kamera abhängig und kann über das Hilfsprogramm VCAM95/ICAM.EXE festgestellt werden.

Returnwerte: --

Beispiel: --

Die Blank Time beginnt mit der fallenden Flanke von HD und wird bis zum Beginn des Video Signals eingestellt wobei in 2*pclk gezählt wird (1 increment = 0.14096 usec). Erst mit dem Ende der blank_time beginnt die Aufzeichnung der Video Daten. blank_time bestimmt auch die Länge des clamp Signals.

2.4.2.8 mvfg_pal (0/1), mfg_pal (0/1)

Synopsis: void mvfg_pal (DWORD pal);

Beschreibung: Festlegen der Videonorm der Kamera.
pal = 0 --> 525 Zeilen (NTSC) oder 128 Zeilen bei Zeilenkameras.
= 1 --> 625 Zeilen (CCIR) oder 256 Zeilen bei Zeilenkameras.

Returnwerte: --

Beispiel: --

2.4.2.9 mvfg_clamp(Clamp), mfg_M437SetClamp (Clamp)

Synopsis: void mvfg_clamp(DWORD dwClamp); // Windows
void mfg_M437SetClamp (DWORD dwClamp); // DOSX

Beschreibung: Einstellen des Clamplevels für das Videosignal das an den Hochgeschwindigkeits Video A/D Wandler (Zusatzbaugruppe M437/M446) angeschlossen ist.
Wertebereich: dwClamp = 0..255, oder 0...?V, default Wert 165 (eingestellt durch mvfg_modcam ())

Returnwerte: --

Beispiel: --

2.4.2.10 mvfg_gain(Gain), mfg_M437SetClamp (Gain)

Synopsis: void mvfg_gain(DWORD dwGain); // Windows
void mfg_M437SetGain (DWORD dwGain); // DOSX

Beschreibung: Einstellen des Verstärkungsfaktors für das Videosignal das an den Hochgeschwindigkeits Video A/D Wandler (Zusatzbaugruppe M437/M446) angeschlossen ist.
Wertebereich: dwGain = 0..255, oder Vout = (??? - dwGain) * Vin, default Wert 153 (eingestellt durch mvfg_modcam ())

Returnwerte: --

Beispiel: --

2.4.2.11 mvfg_get_clamp(void)

Synopsis: DWORD mvfg_get_clamp(void); // Windows
globale Variable: m437_clamp; //DOSX

Beschreibung: Lesen des eingestellten Clamplevels

Returnwerte: DWORD Clamp, Wertebereich 0..255

Beispiel: --

2.4.2.12 mvfg_get_gain(void)

Synopsis: DWORD mvfg_get_gain(void); // Windows
globale Variable: m437_gain; //DOSX

Beschreibung: Lesen des eingestellten Verstärkungsfaktors

Returnwerte: DWORD Gain, Wertebereich 0..255

Beispiel: --

2.4.3 Speicherverwaltung

Die Funktionen mvfg_hstart (), h_start und mvfg_selframe (), mfg_selframe und mvfg_lines(), mfg_lines bestimmen, wie die Zeilen im Bildspeicher abgelegt werden.

mvfg_xchg(), mfg_xchg klappen den Bildspeicher um.

mvfg_input (), klappen den Bildspeicher beim nächsten Bildimpuls um.
mvfg_set_vflag(), mvfg_get_vflag() setzen oder lesen das _fgtv_valid Flag.

Wenn immer nur ein einziges Bild aufgezeichnet und automatisch ein Speicherblock-Tausch durchgeführt werden soll, ist dafür eine Flag Steuerung mit dem globalen Flag DWORD _fgtv_valid vorgesehen.

Das Flag _fgtv_valid wird in der Interrupt Service Routine des vertikalen Syncs (genauer: in der Routine mfg_sync) überprüft, und wenn =0 wieder auf 1 gesetzt und die Funktion mfg_xchg aufgerufen und damit ein Speicherblocktausch durchgeführt.

Das folgende Beispielprogramm ist gleichzeitig ein voll funktionsfähiges Demo Programm für den INSPECTA:

```
/* module       demo.c
====================================
example program for INSPECTA demonstration
====================================*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include <ctype.h>
#include <mvfg.h>

 main (int argc, char * argv[])       /* MAIN PROGRAM
======================================*/
{

printf ("Ergebnis von mfgtest: %d\n",   mfgtest());

mfg_camsel(0);      // select camera 0
mfg_modcam(0x68);   // analog interlace, composite sync
mfg_blank(0x45);    // blank time
mfg_black(0x1c);    /* number of black lines from vertical
                       sync going hi to first visible line */
mfg_pal(1);         /* not NTSC frames

h_start(linelen,  numlin,     interlace,  req_frame); */

h_start(0x2d4,    0x25e,      1,          -1);
vmfg_isr(0x10);
mfg_int(0x410);

init928(0);         // umschalten in s3 mode 103 für 640 x 480 mit
                    // 8 bit pro pixel
mfg_clip(0);        // evtl gesetztes clip window ausschalten
palette928(0);      // s/w palette mit 4 overlay farben für bit 0..1

while (!_kbhit ())  // solange keine Taste gedrückt
```

```
 {

  _fgtv_valid = 0 ;              // setze flag auf speichertausch in nächstem
                                 // vertikalen interrupt
  while( _fgtv_valid == 0 );     // warte bis in service routine zurückgesetzt

/* zur Bildausgabe die Funktion vmfg_put2win benützen
      vmfg_put2win (ptr,     x, y, width, skippix, height, skipline, pitch, ram)*/
      vmfg_put2win (mfg_lin, 0, 0, 640,   0,       480,    0,        0x2d8, 1);
  }
mfg_int(0);         // mvfg-interrupt ausschalten
vmfg_isr(0); // PC Interrupt zurücksetzen
init928(-1);      // select vga character mode before exit

}                                          /* end "main" */
```

Beispiel für Mehrfachaufzeichnung:
Von drei analogen Kameras soll je ein Bild aufgezeichnet und hintereinander abgespeichert werden. Nach dem letzten Bild soll die Aufzeichnung stoppen. Der Speicherblock-Tausch soll dann durchgeführt werden, wenn noch laufende Berechnungen aus der vorangegangenen Bilderserie abgeschlossen sind.

```
...
int status;
mfg_modcam (0x38);  // analoge Kamera, Takt von und Sync zur Kamera
mfg_camsel(0);            // waehle Kamera 0
mfg_selframe(0);          // schreibe ins Bild 0
mfg_startstop(1);         // Einspeichern starten

/* warte bis bit fdv == 0 und bit odd == 1 in mfg_stat()
   d.h. bis zum nächsten Vollbild */

       while (( status = ( mfg_stat() && 0xa00 ) == 0x800 );

/* jetzt wird bild 0 geschrieben */

/* warte bis bit fdv == 0 und bit odd == 1 in mfg_stat()
   d.h. bis zum nächsten Vollbild */

       while (( status = ( mfg_stat() && 0xa00 ) == 0x800 );

/* jetzt ist bild 0 fertig, nächste Kamera und nächstes
   Bild wählen */

mfg_camsel(1);            // waehle Kamera 1
mfg_selframe(1);          // shreibe ins Bild 1

/* warte bis bit fdv == 0 und bit odd == 1 in mfg_stat()
   d.h. bis zum nächsten Vollbild */

       while (( status = ( mfg_stat() && 0xa00 ) == 0x800 );

/* jetzt ist bild 1 fertig, naechste Kamera und nächstes
   Bild wählen */

mfg_camsel(2);            // waehle Kamera 2
mfg_selframe(2);          // shreibe ins Bild 2

/* warte bis bit fdv == 0 und bit odd == 1 in mfg_stat()
   d.h. bis zum naechsten vollbild */

       while (( status = ( mfg_stat() && 0xa00 ) == 0x800 );

/* jetzt ist bild 2 fertig, weiteres Einspeichern stoppen */

mfg_startstop(0);

....

/* jetzt wird die vorhin gespeicherte Sequenz zur weiteren Berechnung
   benoetigt */

mfg_xchg();        // jetzt stehen die Bilder ab *mfg_lin im Speicher
```

Die obige Form der Sequenzbildung hat den Nachteil, daß das der Kameraumschaltung folgende Bild nicht aufgezeichnet werden kann, da das Laden der Zeilen-Tabelle (durch mfg_selframe) nicht innerhalb der vertikalen Dunkelpause abgeschlossen wird.
Die Kamera-Steuerungsfunktionen m(v)fg_multisequence(), m(v)fg_multiframe() vermeiden diesen Effekt, indem die gesamte Abfolge der Aktivitäten im Hintergrund, in der INSPECTA-Interrupt-Service Routine durchgeführt werden.

2.4.3.1 mvfg_black (black_lines), mfg_black (black_lines)

Synopsis: void mvfg_black(DWORD black_lines);

Beschreibung: Über die Funktion wird die Anzahl ungültiger Zeilen vom Start des vertikalen Synchronisationssignals bis zur 1. gültigen Zeile des Bildes festgelegt.

Dieser Wert muß unbedingt **vor** Aufruf der Funktion '**mvfg_hstart()**' gesetzt werden, da er erst mit Aufruf dieser Funktion wirksam wird !

Dieser Wert ist vom Typ der Kamera abhängig und kann über das Hilfsprogramm VCAM.EXE festgestellt werden.

Returnwerte: --

Beispiel: --

mfg_black schreibt die globale variable black_lines, die in der Funktion h_start dazu benützt wird, die Startadresse für diese Zeilen an das Ende des Bildspeichers zu setzen.

2.4.3.2 mvfg_blackend (black_linesend), mfg_blackend (black_linesend)

Synopsis: void mvfg_blackend (DWORD black_linesend);

Beschreibung: Über die Funktion wird die Anzahl der Zeilen festgelegt, die nach der letzten gültigen Zeile bis zum Ende des Bildes (bis zum nächsten vertikalen Synchronisationssignal) von der Kamera geliefert werden und nicht abgespeichert werden sollen (“schwarze Zeilen am Ende des Bildes“).

Dieser Wert muß unbedingt **vor** Aufruf der Funktion '**mvfg_hstart()**' gesetzt werden, da er erst mit Aufruf dieser Funktion wirksam wird !
Dieser Wert ist vom Typ der Kamera abhängig und kann über das Hilfsprogramm PCAM.EXE festgestellt werden.

Returnwerte: --
Beispiel: --

2.4.3.3 mvfg_hstart, h_start (linelen, numlin, interlace, req_frm)

Synopsis: void mvfg_hstart (DWORD linelen, DWORD numlin, DWORD interl, LONG requ_frm);

Beschreibung: Über diese Funktion wird die Geometrie, der Typ und die Anzahl zu erfassender Frames des aufzuzeichneten Videobildes festgelegt.

Parameter: Die Parameter haben im einzelnen folgende Bedeutung:

linelen: Anzahl Pixel / Bildzeile.

numlin: Anzahl von Zeilen / Frame **inkl**. der durch 'mvfg_black()' und mvfg_blacklinesend() definierten Zeilen. INSPECTA zählt alle HSYNC Pulse außerhalb von VSYNC == 0, also auch die „Serration“ und „Equalizing“ Pulse unmittelbar vor oder nach dem VSYNC eines composite Video Signals. Der bei Standard Video Signalen (NTSC/RS-170 bzw. CCIR) einzusetzende Wert ist daher um ca. 20 kleiner als die zu erwartenden 525/625 Zeilen.
Bei Linescan Kameras wird intern ein VSYNC nach 128 bzw. 256 Zeilen erzeugt. Dies wird mit mvfg_pal (0) = 128 oder mvfg_pal (1) = 256 ausgewählt.

interl: Typ des Videobildes. 1 = interlaced, 0 = non interlaced.

requ_frm: Nummer des Speicherposition im Bildspeicher in dem das aufgezeichnete Bild abgelegt werden soll. 0 ist die erste Position..
Wird für 'requ_frm' -1 angegeben, werden soviele Bilder in der Zeilentabelle belegt, bis alle 2048 Zeilen (8192 für INSPECTA) je Kanal belegt sind oder soviele Bilder wie in den reservierten Bildspeicher hineinpassen, definiert. (rotating mode)
Bei ‘requ_frm’ < -1 werden nur soviel Bilder definiert wie requ_frm-1 angibt.
Für INSPECTA-1 stehen je Kanal 2048 Zeilen, bei INSPECTA 8192 Zeilen je Kanal zur Verfügung.

Die hier angegebenen Parameter können über das Hilfsprogramm VCAM95 oder ICAM.EXE eingestellt werden.

Returnwerte:

Beispiel: /* Geometrie eines Frames einer PULNIX TM9700 im dig. Modus festlegen. Das Videosignal ist non interlaced, es soll nur in Frame 0 geschrieben werden (1 Bild). */

```
mvfg_hstart( 0x300, 0x20d, 0x0, 0x0 );
```

Verarbeitung interner Variablen:

Die Aufrufparameter linelen, numlin, interlace und req_frm werden in globale Variablen abgelegt.

Der Aufrufparameter numlin bezeichnet die Gesamtzahl aller Zeilen eines Bildes, also auch schwarze Zeilen zu Beginn und am Ende eines Bildes, die nicht abgespeichert werden.

Die globale Variable dword pel_frm wird berechnet (= linelen*(numlin-black)), desgleichen die globale Variable dword frame_nr ((=[mfg_len]/pel_frm), Anzahl Bilder pro 1/4/16MB)

h_start benützt zusätzlich die glob. Variable black_lines, die dafür sorgt das nicht aufzuzeichnende (z.B.: schwarze) Zeilen am Anfang des Bildes an das äußerste Ende des Bildspeichers (=[mfg_len] - linelen) gespeichert werden. Die Variable dword black_lines wird durch die Funktion mfg_black geschrieben, jedoch erst mit h_start ausgeführt.

h_start benützt zusätzlich die glob. Variable black_linesend, die dafür sorgt das nicht aufzuzeichnende (z.B.: schwarze) Zeilen am Ende des Bildes an das äußerste Ende des Bildspeichers (=[mfg_len] - linelen) gespeichert werden. Die Variable dword black_linesend wird durch die Funktion mfg_blacklinesend geschrieben, jedoch erst mit h_start ausgeführt.

2.4.3.4 mvfg_selframe (framenr), mfg_selframe (framenr)

Synopsis: void mvfg_selframe (LONG nr);

Beschreibung: mfg_selframe (nr) ist eine Kurzform von mvfg_hstart (). Als Aufrufparameter wird nur die Bildposition im Bilspeicher (req_frm bei mvfg_hstart) angegeben. Alle anderen Aufrufparameter von mvfg_hstart () werden aus den zugehörigen globalen Variablen entnommen. Der Wert von 'nr' darf nicht größer sein als:
0 <= nr <= MaxFrames.
MaxFrames wird durch die Funktion 'mvfg_hstart()' errechnet.

Bei 'nr' == -1 wird in den rotierenden Modus geschalten (siehe 'mvfg_hstart()').

Returnwerte: --

Beispiel: --

mfg_selframe benutzt die variablen linelen, numlin, interlace und schreibt die variable dword req_frm. Zur Ausführung wird die Funktion h_start aufgerufen.

2.4.3.5 mvfg_PhysBuffer (*framestruc), mfg_PhysBuffer (*framestruc)

Synopsis: void mvfg_PhysBuffer (*framestruc);

Beschreibung: mfg_ PhysBuffer(*framestruc) enhält in der Struktur *framestruc alle Parameter zur Definition des Bildformats.

*framestruc:
+0 physical buffer address (byte address)
+4 physical buffer length
+8 single/double buffer: 0/1
+12 linelen
+16 numlin
+20 interlace 0/1
+24 req_frame -1, 0..frame_nr
+28 blacklines >=0<numlin
+32 blacklinesend >=0<(numlin-blacklines)
+36 seq_color 0,1,2

double_buffer:
0: only one image buffer
1: two image buffers, length = physical_buffer/2

seq_color:
0: single plane b&w
1: three planes for three b&w or one RGB camera
2: two planes interlaced into each other, for two-tap sensors like SONY XC-7500

Returnwerte: --
Beispiel: --

2.4.3.6 mvfg_input (DWORD timeout)

Synopsis: LONG mvfg_input (DWORD timeout);

Beschreibung: Die Funktion wartet auf den nächsten VSYNC von einem oder sovielen Vollbildern, wie duch die Funktion mvfg_hstart mit parameter req_frm <0 angefordert wurden. Dann wird der Bildspeicher „umgeklappt“, bzw. bei INSPECTA der Pointer (byte *) plMvfgDatPtr MVFGD_BUF_POINTER] auf die neue Bildspeicherhälfte gesetzt.

Konnte der VSYNC nach Ablauf von 'timeout' Mikrosekunden nicht gefunden werden, so wird mit einer entsprechenden Fehlermeldung an den Aufrufer zurückgekehrt. Es wird der interne Millisekunden Timer von Windows verwendet. Der Parameter DWORD timeout wird aus Kompatibilitätsgründen in Mikrosekunden angegeben.

Das Umklappen des Bildspeichers wird in der vertikalen Interrupt Service-Routine durchgeführt. D.h. das eingelesene Bild wird asynchron zum laufenden Programm in den Hauptspeicher übertragen !

Returnwerte:

== 0	Bild erfolgreich eingelesen.
== EMVFG_TIMEOUT	Timeout bei einlesen des Bildes.

Beispiel:

```
if ( mvfg_input ( 0x100000 ) == EMVFG_TIMEOUT )
{
wsprintf( acPuffer, "Timeout bei lesen des nächsten Bildes !" );
...
}
```

Bemerkung: Diese Funktion steht nur unter Windows zur Verfügung

2.4.3.7 mvfg_inputEx(int flag, DWORD timeout)

Synopsis: LONG mvfg_inputEx(int flag, DWORD timeout)

Beschreibung: ‚mvfg_inputEx()' erweitert die Funktion ‚mvfg_input()' um die bei ‚mvfg_set_vflag()' beschriebenen Kontrollmöglichkeiten zum Bildeinzug. Während die Funktion auf den Einzug der Bilder wartet, wird der zugeordnete Thread suspendiert, so daß während Wartezeit keine Prozessorzeit verbraucht wird.

int	flag	Bedeutung wie bei ‚mvfg_set_vflag()'
DWORD	timeout	Dauer nach der die Funktion zum Aufrufer zurückkehrt, wenn Bildsignale fehlen. **Bei Einzug von mehreren Bildern, auf ausreichend dimensioniertes Timeout achten!** Der Wert wird in [µs] angegeben.

Returnwert:

== MVFG_OK	Bild erfolgreich eingelesen
== EMVFG_TIMEOUT	Timeout bei einlesen des Bildes

Beispiel:

```
// Einzug einer Sequenz von 5 Frames
mvfg_selframe ( -6 ); // Zeilentabelle des Framegrabbers
                      // für 5 Bilder aufsetzen (nur einmal notwendig).
.
.
.

Mvfg_inputEx( 3, 1000000 );          // Bilder einlesen
```

Bemerkung: Diese Funktion steht nur unter WiNT/2K zur Verfügung

2.4.3.8 mvfg_get_event (DWORD event_id)

Synopsis: Handle mvfg_get_event(DWORD event_id)

Beschreibung: mvfg_get_event() liefert den Handle eines Frame-Grabber-Events zurück. Das Frame-Grabber-Event wird vom Treiber des Frame Grabbers bedient und signalisiert bestimmte Ereignisse/Zustände beim Bildeinzug.

Das Events ist ein „manual reset event object“, d.h. das Event wird vom Frame Grabber bei Eintreten des jeweiligen Ereignisses zwar gesetzt, aber nicht von diesem bei Ende des Ereignisses rückgesetzt. Dies liegt in der Verantwortung des Anwenders.

Das gewünschte Event wird über die Event-ID bestimmt.

Folgende Event-ID’s sind definiert:

ID	Beschreibung
MVFGEVENT_VALID_FLAG	Das Event wird gesetzt (signaled), sobald ein komplettes Bild vom Frame Grabber eingezogen wurde.

Returnwert:

!=	NULL	Handle des Events
==	NULL	Handle nicht verfügbar

Beispiel:

```
// Einzug eines Frames mit Hilfe des Valid-Flag Events:

// Handle des Valid-Flag Events holen
hValidFlagHandle = mvfg_get_event( MVFGEVENT_VALID_FLAG );

ResetEvent( hValidFlagHandle );     // Valid-Flag Event rücksetzen

mvfg_set_vflag( 0 );            // Einzug des nächsten Frames anstoßen

// Warten bis der Frame verfügbar ist, 1 s Time Out
WaitForSingleObject( hValidFlagHandle, 1000 );
```

Bemerkung: Diese Funktion steht nur unter WiNT/2K zur Verfügung

2.4.3.9 mvfg_set_vflag (int flag);

Synopsis: void mvfg_set_vflag (int flag);

Beschreibung: Die Funktion setzt das _fgtv_valid flag auf 0. Das Flag wird bei jedem Auftreten eines VSYNC überprüft und die fogende Aktivität durchgeführt:

int flag = 0 : Der Bildspeicher wird „umgeklappt“, bzw. bei INSPECTA der Pointer (byte *) plMvfgDatPtr MVFGD_BUF_POINTER] auf die neue Bildspeicherhälfte gesetzt. int flag wird auf 1 gesetzt.

int flag = 1 : keine Aktivität bezüglich Bildspeicher oder Aufzeichnungsprozess.

int flag = 2 : Der Bildspeicher wird „umgeklappt“, bzw. bei INSPECTA der Pointer (byte *) plMvfgDatPtr MVFGD_BUF_POINTER] auf die neue Bildspeicherhälfte gesetzt. int flag bleibt auf 2, und damit wird beim nächsten VSYNC wieder „umgeklappt“.

int flag = 3 : Der Bildspeicher wird nach sovielen Vollbildern wie durch die Funktion mvfg_hstart mit parameter req_frm <0 angefordert wurden umgeklappt“, bzw. bei INSPECTA der Pointer (byte *) plMvfgDatPtr MVFGD_BUF_POINTER] auf die neue Bildspeicherhälfte gesetzt. int flag wird auf 1 gesetzt.

int flag = 4 : Die Aufzeichnung wird nach sovielen Vollbildern wie duch die Funktion mvfg_hstart mit parameter req_frm <0 angefordert wurden gestoppt. int flag wird auf 1 gesetzt.

Da die Aktivitäten asynchron im Hintertgrund erfolgen, sollte 'mvfg_set_vflag()' erst dann aufgerufen werden, wenn die aktuellen Bilddaten nicht mehr benötigt werden.

Returnwerte: --

Beispiel: Siehe Funktion 'mvfg_get_vflag ()'.

Bemerkung: Diese Funktion steht nur unter Windows zur Verfügung , unter DOSX wird die globale Variable DWORD _fgtv_valid direkt verwendet.

2.4.3.10 mvfg_get_vflag (int iFlag);

Synopsis: LONG mvfg_get_vflag (int iFlag);

Beschreibung: Lesen des aktuellen Zustands des Valid Flags.

Returnwerte: == 0 Aufnahme läuft noch.
== 1 Aufnahme beendet.

Beispiel
```
mvfg_set_vflag ( MVFG_VALID ); // Speicherblocktausch vorbereiten.

// Warten bis Speicherblocktausch erfolgt ist.

while ( !mvfg_get_vflag ( MVFG_VALID ));
```

Bemerkung: Diese Funktion steht nur unter Windows zur Verfügung , unter DOSX wird die globale Variable DWORD _fgtv_valid direkt verwendet.

2.4.3.11 mvfg_xchg (), mfg_xchg ()

Synopsis: void mvfg_xchg (void);

Beschreibung: Die Funktion schaltet **sofort** die INSPECTA Speicherblöcke um.

Returnwerte: --

Beispiel:
```
/* Warten auf nächsten VSYNC damit der Speicherblocktausch nicht mitten
im Bild durchgeführt wird  */
while ( !(mvfg_stat ( void )  & MVFGS_FRAME_VALID ));

/* Speicherblöcke umschalten          */

mvfg_xchg();
```

2.4.3.12 mvfg_ActualDmaPointer (), mfg_ ActualDmaPointer()

Synopsis: DWORD mvfg_ ActualDmaPointer(void);

Beschreibung: Die Funktion liest die aktuelle Schreibposition des PCI-DMA’s und rechnet diese in die nächstniedrige Zeilennummer um.

Returnwerte: aktuelle Schreibposition modulo linelength.

Beispiel: --

Anmerkung: diese Funktion steht ab Rev. 1.51 zur Verfügung. Für Inspecta-4 und req_frm = -1, -2, -3, .. –n (mvfg_selfrm() und einige Hochgeschwindigkeitskameramodi ist der Rückgabewert konstant 1.

2.4.3.13 m(v)fg_DefineNextImage (DWORD ImageNr, DWORD ShutterTime, DWORD DoubleBuffer)

Synopsis: DWORD m(v)fg_DefineNextImage(DWORD ImageNr, DWORD ShutterTime, DWORD DoubleBuffer);

Beschreibung: Die INSPECTA -2 Zeilentabelle wird mit neuen Zeilenadressen geladen, sodaß das nächste Bild an die durch die ImageNr definierte Position geschrieben wird. DoubleBuffer 1/0 aktiviert oder deaktiviert den durch Hardware ausgelösten Speicherblocktausch.

Für Kameras mit asynchronem Shutter wird die durch Hardware bestimmte Belichtungszeit in Einheiten von Zeilenzeiten mit dem Parameter ShutterTime eingestellt. ShutterTime 0 löst einen asynchronen Belichtungsvorgang mit einer durch die Kamera bestimmten Belichtungszeit aus.

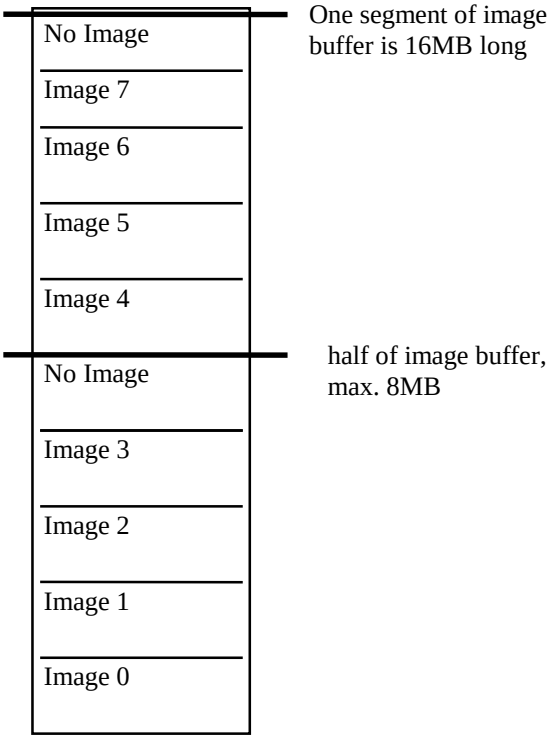
ImageNr bestimmt die Position des Bildes im Bildspeicher. Der Returnwert ist der Offset im Bildspeicher für die gewählte Bildnummer. ImageNr == -1 gibt als Returnwert den Index des letzten in den aktuellen Bilspeicher passenden Bildes zurück.

Aufbau des Bildspeichers:
Der Bildspeicher ist in 16MB Segmente aufgeteilt. Ein Segment kann auch kleiner als 16MB sein, seine physikalische Anfangsadresse plus Länge darf eine 16MB Segmentgrenze nicht überschreiten. Es gibt soviele 16MB Segmente wie in die Gesamtlänge des Bildspeichers hineinpassen.

DoubleBuffer = 1 teilt den Bildspeicher in zwei gleiche Hälften. Unabhängig von der Gesamtlänge des Bildspeichers wird dann nur **ein** Segment mit max. 16MB benützt. Typisch wird ein relativ kleiner Bildspeicher (z.B.: 4MB) mit ImageNr.: 0 aufgerufen.
DoubleBuffer = 1 wird vorzugsweise mit synchronen Kameras benützt wenn die Adressumschaltung im Hintergrund bei minimalster CPU-Belastung (wenige usec) erfolgen soll.

Verteilung der Bilder im Bildspeicher:
Für ein Bild wird (numlin * linelen) Bildspeicher ab physikalischem Beginn reserviert. Es werden soviele Bilder reserviert wie in die Hälfte des Bildspeichers oder in 8MB passen. Der Speicher zwischen dem letzten vollständigen Bild und der Bildspeicherhälfte oder 8MB wird nicht benützt.

Ist der Bildspeicher länger als 16MB widerholt sich die Verteilung wie im u.g. Beispiel, das nächste Bild mit der ImageNr = 8 beginnt am Bildspeicheroffset 16MB.



Returnwerte: Offset im Bildspeicher für die gewählte Bildnummer oder Index des letzten in den Bildspeicher passenden Bildes bei ImageNr == -1.

Beispiel: --

Anmerkung: Die Verarbeitungszeit beträgt ca. 0.5 usec pro Zeile, d.h.: bei einem Bild mit numlin = 508 dauert die Funktion ca. 1msec. Bei DoubleBuffer = 1 verdoppelt sich die Zeit. Bei Shutter Zeiten >0 muß pro Increment (Zeilenzeit) ebenfalls 0.5 usec hinzugerechnet werden.
Wenn die Aufrufparameter gegenüber dem letzten Aufruf nicht geändert werden, wird keine Zeit verbraucht .
m(v)fg_DefineNextImage steht ab Rev. 1.77 nur für INSPECTA zur Verfügung

2.4.4 Kamera-Kontrolle

Die folgenden Funktionen steuern die Belichtung der Bilder. Kürzere Belichtungszeit als eine Bildzeit wird im folgenden auch „Shuttern“, Belichtungszeiten die mehr als eine Bildzeit lang sind werden „Integration“ genannt.

Die Funktionen mvfg_photo(), mfg_photo steuern das „Shuttern“, mvfg_integration () die „Integration“

Die Bildaufnahme kann entweder vom Anwendungsprogramm gesteuert werden oder im Hintergrund (mit der INSPECTA Interrupt Service Routine) anhand vor „Descriptoren“ gesteuert werden. Dies kann entweder nach Ablauf einer Video Zeile (=64 usec) oder eines Video (Halb)Bildes (= 16 .. 50 msec) passieren.

2.4.4.1 mvfg_startstop (DWORD flag); mfg_startstop (DWORD flag)

Synopsis: void mvfg_startstop (DWORD flag);

Beschreibung: Die Funktion startet bzw. stopt Aufnahmen durch den INSPECTA

Mögliche Werte für 'flag':

MVFG_STOP = 0 Aufnahme anhalten.
MVFG_START = 1 Aunahme sofort starten und den internen Schreibzeiger zurücksetzen sodaß die nächsten Zeilen ab Anfang des Bildspeichers aufgezeichnet werden.
MVFG_START = 2 oder 3 Aufnahme beim nächsten Vollbild (zwei Halbbilder bei interlace Kameras) starten und den internen Schreibzeiger zurücksetzen sodaß die nächsten Zeilen ab Anfang des Bildspeichers aufgezeichnet werden.

Returnwerte: --

Beispiel: --

2.4.4.2 mvfg_stat (), mfg_stat ()

Synopsis: DWORD mvfg_stat (void);

Beschreibung: Über die Funktion wird der akt. Status des INSPECTAS ermittelt.

Returnwerte: mfg_stat () liefert vier Byte Status Code.

Bits:	0=MC0	camera special fct. control 0
	1=MC1	camera special fct. control 1
	2=MC2	camera special fct. control 2 / VINIT
	3=MC3	camera special fct. control 3 / integration

4=CAM0 select one of two a/d input triples
5=RGBX0 data width
6=RGBX1 data width
7=RGBX2 data width

8=LDV 1=linedata valid
9=FDV 1=frame data valid (analog or digital)
10=PHO 1=Photo has been started and is not ready
0=Photo is ready (or not started)
11=ODD 1=Odd field
12=pixel clock/4 (>= rev. 4): internal pixel clock/4
13= undefined (>= rev. 4)
14=FDV digital 1=digital camera frame data valid
15=undefined

16 1= connects video red data buffer to red channel
17 1= connects video green data buffer to red channel
18 1= connects video blue data buffer to red channel
19 1= connects digital video data buffer to red channel
20 1= interrupt enabled
21 undefined
22 undefined
23 DIS_WR 1= disable writing to fifo

24=AM0 camera mode bit 0
25=AM1 camera mode bit 1
26=ISRC0 interrupt source 0
27=ISRC1 interrupt source 1
28=DIS_EOL disable end of line processing if set
29=PAL 525/625 lines
30=STEST1 camera mode bit 2
31=STEST camera mode bit 3

ISRC [0..2] = ; interrupt occurs on
0/0 ; hor. freq/128
1/0 ; disable write to fifo
0/1 ; falling edge of VSYNC
1/1 ; falling edge of HSYNC

2.4.4.3 mfg_fdvhi ()

Synopsis: DWORD mfg_fdvhi (void);
Beschreibung: Diese Funtion wartet auf die steigende Flanke von FDV (VSYNC)

Rückgabewert: 0 wenn o.K, -1 wenn timeout (ca. 0.5sec)

Remark: DOSX only

2.4.4.4 mfg_fdvlo ()

Synopsis: DWORD mfg_fdvlo (void);
Beschreibung: Diese Funtion wartet auf die fallende Flanke von FDV (VSYNC)
Rückgabewert: 0 wenn o.K, -1 wenn timeout (ca. 0.5sec)
Remark: DOSX only

2.4.4.5 mvfg_photo (time), mfg_photo (time)

Synopsis: LONG mvfg_photo (LONG time);

Beschreibung: Die Funktion steuert den (elektronischen) Verschluß einer Kamera.

mfg_photo () benützt zum timen der Verschlußzeit bei variablen Verschluß-zeiten den HSYNC, den Pixeltakt oder einen 1us Timer als Zeitbasis.

In der folgenden Tabelle ist die Verwendung der unterschiedlichen Zeitbasen in Abhängigkeit vom Typ des Inspectas und der Kamera angegeben. (SZ = Verschlusszeit)

Kameratyp	Inspecta-2/3	Inspecta 4D	Inspecta 4C
Linescan	Keine SZkontrolle möglich	SZ=1/Pixelclock* time $SZ_{max.}$=1/Pixelclock* 65530	SZ= 1µs * time $SZ_{min.}$= 2µs $SZ_{max.}$= 8ms
Matrix Kamera ohne LDV beim Warten auf Trigger, z.B.: Adimec 1000M Teli CSB4000CL Basler Ax02k JAI-M4	Keine SZkontrolle möglich	SZ=1/Pixelclock* time $SZ_{max.}$=1/Pixelclock* 65530	SZ= 10us * time $SZ_{min.}$= 20µs $SZ_{max.}$= 80ms
Matrix Kamera mit LDV beim Warten auf Trigger	SZ = Zeilenzeit * time	SZ = Zeilenzeit * time	SZ = Zeilenzeit * time

Returnwerte: 0

Beispiel: /* Warten auf einen Triggerimpuls. Bei Eintreten des Ereignisses
soll der Verschluß der Kamera, die im Shuttermodus 4 läuft, betätigt werden.
Wenn das Bild fertig ist, soll der Speicherblocktausch durchgeführt werden.
*/

```
/* warte auf Trigger, Bit 0 vom Opto-Input */
while ( mvfg_ppin() & 0x01 );
mvfg_photo (40);                /* Verschluß steuern  */
return mvfg_input ( timeout );  /* zurückkehren wenn Bild fertig
                                   oder Timeout */
```

2.4.4.6 mvfg_ScanPeriod (ScanPeriod), mfg_ScanPeriod (ScanPeriod)

Synopsis: void mvfgScanPeriod(DWORD ScanPeriod);

Beschreibung: Stellt die Frequenz das Start of Scan Signals für eine linescan Kamera ein. Der Parameter ScanPeriod zählt in Einheiten von Pixel-Takten.

ScanPeriod ist größer oder gleich der Zeilenlänge der Kamera und kleiner als 65530.

Inspecta-2/3 und Inspecta-4A benützen den Wert ScanPeriod = 128 (und m(v)fg_pal =0) für den Betrieb mit einem externen SOS Signal. Bei ScanPeriod = 128 wird der interne SOS generator gestoppt.

Inspecta-4D/C haben dafür ein eigenes Kontrollbit, dass mit der Funktion mvfg_linescan () bedient wird.

Returnwerte: --

Beispiel: --

mfgScanPeriod schreibt die globale Variable start_scan.

2.4.4.7 mvfg_contWrite (DWORD NrOfFrames, DWORD Circular);

Synopsis: mvfg_contWrite (DWORD NrOfFrames, DWORD Circular);

Beschreibung: Die Funktion startet die Aufzeichnung von Bildsequenzen bis zur angegebenen Anzahl von Frames. Bei Halbbildkameras werden doppelt soviele Fields aufgezeichnet.
DWORD Circular = 0: einmalige Aufzeichnung, _fgtv_valid = 1 nach Beendigung.
DWORD Circular = 1: ununterbrochene Aufzeichnung, stop mit mvfg_startstop (0), Wiederstart durch erneuten Aufruf. Die globale Variable (frm_cnt) bzw. DatPointer[MVFGD_FRM_CNT] zählt die aufgezeichneten Frames. Bei frm_cnt = 0 ist eine Sequenz fertig.

Returnwerte: --

Beispiel: --

Bemerkung: Treiberversion > 1.65 und INSPECTA Hardware Rev. >4, IMP Nr. >317 notwendig. Multiplane Kamera Modi sind nicht möglich. Alle Videozeilen außerhalb des vertikalen Sync-Impulses werden aufgezeichnet. Halbbilder werden nicht eingekämmt. Mit mvfg_blank (blanktime) können die schwarzen Pixel zu Beginn der Zeile unterdrückt werden. Vorherige Intialisierung mit mvfg_contWriteInit (DWORD PhysAddr) notwendig.

2.4.4.8 m(v)fg_Snap (DWORD mode, DWORD stop/exchange);

Synopsis: m(v)fg_Snap (DWORD mode, DWORD stop/exchange);

Beschreibung: Die Funktion startet die Aufzeichnung eines Bildes, erlaubt einem externen Triggersignal am Eingang 0 der opto-gekoppelten Eingänge die Auslösung eines Belichtungsvorgangs und stoppt und oder tauscht den Speicherblock aus. Parameter mode und stop/exchange haben folgende Auswirkung:

mode:

= 0:	*start grab on next vsync, stop or toggle and stop then, _fgtv_valid = 1 (not implemented.)*
= 1:	*start grab on next vsync, stop or toggle and stop then, _fgtv_valid = 1 (not implemented.)*
= 2:	*start toggling on next even field, continue until vsync after requested stop, _fgtv_valid = 1 (not implemented.)*
= 3:	*start random shutter, stop or toggle and stop when ready; _fgtv_valid = 1*
= 4:	*enable external shutter on opto-in bit 0 high pulse, stop or toggle and stop when ready, _fgtv_valid flag = 1*
= 5:	*grab multiple images as long as external signal on opto-in bit 0 is high. Use mvfg_selframe (# frames) to define maximum number of images.*
= 8:	*Clear enable external trigger. Used if a previous mvfg_Snap (4, x) is to be terminated without an external trigger signal..*
= 10:	*For linescan cameras: issue mvfg_photo() when opto in 0 activated to clear dma pointer, e.g. to start next image with first line.*
= 11:	*For linescan cameras: read ActualDmaPointer when opto in 0 activated and write result to Grabber Variable MVFGD_TRIG_DMA_POINTER*

stop/exchange:

= 0:	stop if action done
= 1:	stop and exchange (except mode 2) if action done

Returnwerte: --

Beispiel: **--**

Bemerkung: Treiberversion >= 1.77 und INSPECTA Hardware Rev. >4, IMP Nr. >=383 notwendig. Bei Verwendung von mvfg_Snap zusammen mit **Inspecta-2/3** sind die opto gekoppelten Ausgänge (**Funktion mvfg_ppout()) ncht mehr verwendbar**.

2.4.4.9 m(v)fg_Snapx (DWORD mode, DWORD stop/exchange, DWORD photo_time);

Synopsis: m(v)fg_Snapx (DWORD mode, DWORD stop/exchange, DWORD photo_time);

Beschreibung: Die Funktion erlaubt einem externen Triggersignal am Eingang 0 der opto-gekoppelten Eingänge die Auslösung eines Belichtungsvorgangs und startet oder stoppt die Aufzeichnung eines Bildes, und oder tauscht den Speicher-block aus. Parameter mode, stop/exchange und photo_time haben folgende Auswirkung:

mode:

= 0:	*start grab on next vsync, stop or toggle and stop then, _fgtv_valid = 1 (not implemented.)*
= 1:	*start grab on next vsync, stop or toggle and stop then, _fgtv_valid = 1 (not implemented.)*
= 2:	*start toggling on next even field, continue until vsync after requested stop, _fgtv_valid = 1 (not implemented.)*
= 3:	*start random shutter, stop or toggle and stop when ready; _fgtv_valid = 1*
= 4:	*enable external shutter on opto-in bit 0 high pulse, stop or toggle and stop when ready, _fgtv_valid flag = 1*
= 5:	*grab multiple images as long as external signal on opto-in bit 0 is high. Use mvfg_selframe (# frames) to define maximum number of images.*
= 8:	*Clear enable external trigger. Used if a previos mvfg_Snapx (4, x, y) is to be terminated without an external trigger signal..*
= 10:	*For linescan cameras: issue mvfg_startstop (2) when opto in 0 activated to clear dma pointer, e.g. to start next image with first line.*

stop/exchange:

= 0:	stop if action done
= 1:	stop and exchange (except mode 2) if action done

photo_time shutter time, same definition as in m(v)fg_photo ().

Returnwerte: --

Beispiel: --

Bemerkung: Treiberversion >= 1.77 und INSPECTA Hardware Rev. >4, IMP Nr. >=383 notwendig. Bei Verwendung von mvfg_Snap zusammen mit **Inspecta-2/3** sind die opto gekoppelten Ausgänge (**Funktion mvfg_ppout()) ncht mehr verwendbar**.

2.4.4.10 mvfg_GrabberSelect (DWORD number);

Synopsis: DWORD mvfg_GrabberSelect (DWORD number);

Beschreibung: mvfg_GrabberSelect wählt einen von vier INSPECTAS in einem WinNT PC aus. Alle folgenden Aufrufe werden zum selektierten INSPECTA geleitet.

Die Funktion mvfg_datpnt () gibt den Pointer für den Datenbereich des selektierten INSPECTA zurück.

Die Bildspeicher Startadresse wird definiert in der WinNT Registry durch den „MemStartPage“ Parameter im Schlüssel: [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\MpfgNtx] wobei MpfgNtx (x=0..3) der Treiber für den zugehörigen INSPECTA ist. (siehe auch: 2.1.3)

number: = 0..3: einer von vier INSPECTAS

Returnwerte: die Nummer des zuletzt selektierten INSPECTAS, oder –1 wenn der selektierte INSPECTA nicht präsent ist.

Beispiel: --

Bemerkung: Treiber Version >= 2.12 und INSPECTA hardware rev. >4, IMP Nr. >=445 werden benötigt.

2.4.4.11 mvfg_Linescan ()

Synopsis: void mvfg_Linescan (DWORD dwScanRate, DWORD dwExpTime, DWORD dwEnaEnc, DWORD dwDivider);

Beschreibung: Alle Einstellungen zum Betrieb einer Zeilenkamera mit dem Inspecta-4D (RS-644 parallel) oder dem Inspecta-4C (Kameralink) können mit dieser Funktion beinflusst werden.

dwScanRate: teilt den Pixeltakt der Kamera durch diesen Wert und steuert damit die Horizontalfrequenz der Zeilenkamera, solange dwEnaEnc == 0 ist. Der Wert muß größer als die Zeilenlänge der Kamera, und kleiner als 65535 sein.

dwExpTime:

Kameratyp	Inspecta-2/3	Inspecta 4D	Inspecta 4C
Linescan	Keine SZkontrolle möglich	SZ=1/Pixelclock* time SZ_{max}.=1/Pixelclock* 65530	SZ= 1µs * time SZ_{min}.= 2µs SZ_{max}.= 8192µs

dwEnaEnc: Deaktiviert oder aktiviert einen externen Encoder.

= 0 Encoder ist inaktiv

= 1 einphasiges Encodersignal auf Bit 2 des opto Eingangs geteilt durch den Teiler: *dwDivider* 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: *dwDivider* 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: *dwDivider* 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.

= 1..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.

dwDivider: teilt den Encodertakt durch diesen Wert und löst je nach Richtung eine Zeile in der Kamera aus. Der Teilerfaktor ist 1+*dwDivider* für *dwDivider* = 0...255.

Returnwerte: --

Beispiel: --

Bemerkung: 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: dwExpTime 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-4DC bestimmt ist (je nach Kamerahersteller z.B.: „level controlled exposure time“), wird der Parameter: dwExpTime 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 Optoin 0 und Funktionsaufruf mvfg_photo (PhotoTime)) überschreibt PhotoTime den Wert „dwExpTime“ der dann pro Bild auch geändert werden kann.

Für die Belichtungszeit bei den verschiedenen Typen der Inspecta Serie gilt:

Kameratyp	Inspecta-2/3	Inspecta 4D	Inspecta 4C
Linescan	Keine SZkontrolle möglich	SZ=1/Pixelclock* time SZ_{max}.=1/Pixelclock* 65530	SZ= 1µs * time SZ_{min}.= 2µs SZ_{max}.= 8192µs

Benötigt: Treiberversion >= 2.75 und Inspecta-4D/C Hardware.

2.4.5 Darstellungsfunktionen

Die leistungsfähigen Darstellungsfunktionen benötigen VGA Karten, die mit einem S3 Grafik-Beschleuniger ausgestattet sind. Alle vorgesehenen Auflösungen und Farbtiefen kommen mit 1MB Video Ram aus.

Dabei wird entweder die Grafik Engine des S3 Chips, oder der direkte Zugriff auf den Video Speicher benützt.

Die Grafik Engine erlaubt Darstellungen mit Overlays, Vergrößern oder Verkleinern, Hardware BitBlts, Polylines mit Hardware Bresenham, Clip und Fill Funktionen.

Bilddarstellung mit direktem Speicherzugriff erlaubt höchste Datentransferraten (bis 40 MB /sec.) und damit Echtzeitdarstellung mit mehr als 25 Vollbildern pro Sekunde oder Farbdarstellung mit 24 Bit Farbe. Die Bilder können in Fenstern mit wählbarer Größe und Position dargestellt werden.

2.4.5.1 init928 (gmode)

Beschreibung: Der Grafikcontroller wird für die angeführten Auflösungen und Farbtiefen initialisiert. Für alle Modi ist maximal 2MB Video Ram ausreichend.

Parameter:

```
gmode == 0  for 640 x 480 x 8 resolution
      == 1  for 800 x 600 x 4 resolution
      == 2  for 800 x 600 x 8 resolution
      == 3  for 1024 x 768 x 4 resolution
      == 4  for 1024 x 768 x 8 resolution
      == 5  for 640  x 480 x 24/32 resolution
      == 6  for 640  x 480 x 15 resolution
      == 7  for 1280  x 1024 x 8 resolution
      == -1 VGA mode 3 (80 x 25 color characters)
```

Beispiel:

```
main ();
{
init928 ( 0 );          /* Init to 640*480*8bit */

...                     // do frame grabbing & image processing here

/* end of programme */
vmfg_isr ( 0 )          // switch off interrupt & restore vector
init928 ( -1 );         // select vga mode 3
}
```

2.4.5.2 alette928 (palette)

Beschreibung: Der ‘palette’ Parameter hat folgende Bedeutung:

```
palette == 0 -> linear 8 bit greyscale with lsb two bits overlay:
                overlay = 0 -> black
                overlay = 1 -> red
                overlay = 2 -> green
                overlay = 3 -> white
palette == 1 -> 3/3/2 red/green/blue 256 color palette
palette == 2 -> expanded 8 bit greyscale
```

Wenn der palette Parameter 0 ist, wird eine Palette geladen. die in Incrementen von vier Pixel-Wertigkeiten um jeweils eine Graustufe erhöht. (max. 63 = weiß).
Beispiel:

pixel value	VGA output
0	schwarz overlay
1	rot overlay
2	grün overlay
3	weiß overlay
4	1
16	4
255	63

Die Ausgabefunktionen, die die Grafik-Engine des 928 benützen (vmfg_putxwin, mfg_c, mfg_fill, mfg_line, mfg_pblt) enthalten einen Aufruf-Parameter (RAM), der bestimmt, ob das Bild oder das Overlay geschrieben werden soll.

2.4.5.3 mfg_clip (onoff)

Beschreibung: mfg_clip schaltet die Clip - Grenzen ein- oder aus. Wenn noch nie ms3_setclipwindow aufgerufen wurde, muß mfg_clip vor Benützung der folgenden Grafik Funktionen ausgeschaltet sein. (mfg_clip(0))

Beispiel:
```
mfg_clip(0);   // clip-window ausschalten
mfg_clip(1);   // clip-window einschalten
```

2.4.5.4 ms3_setclipwindow (x, y, w, h)

Beschreibung: ms3_setclipwindow setzt ein Clip- Fenster vom Punkt x, y (0, 0 ist linke obere Ecke des Schirms) in der Breite w (width) und Höhe h (height).

Beispiel: ms3_setclipwindow (x, y, w, h);

2.4.5.5 mfg_setcolor (foreground, background)

Beschreibung: mfg_setcolor legt für die Darstellungsfunktionen mfg_fill(), mfg_c() und ms3_line die Vordergrund- und Hintergrundfarbe fest. Die Farben hängen von der zuvor gewählten Palette ab:

2.4.5.5.1 Palette 0 (linear 8 bit greyscale with lsb two bits overlay):

Bein Schreiben ins Overlay bestimmen die niedrigsten 4 Wertigkeiten der Parameter foreground (und bei mfg_c() auch der Parameter background) die Overlay-Farben:
foreground = 0: transparent
foreground = 1: rot
foreground = 2: grün
foreground = 3: weiß

Beim Schreiben ins Bild werden mit den höheren 6 Bits der Parameter die Farben festgelegt:
foreground = 0: schwarz
grau
foreground = 252: weiß

2.4.5.5.2 Palette 1 (256 color palette):

Es ergeben sich folgende Werte:
foreground = 0: schwarz
foreground = 255: weiß. Die Farben (von Rot bis Violett) sind mit den entsprechenden Zwischenwerten darzustellen.

2.4.5.5.3 Palette 2 (expanded 8 bit greyscale):

oreground = 0: schwarz
foreground = 255: weiß. Werte dazwischen führen zu Grauwerten.

2.4.5.6 mfg_fill (x, y, w, h, ram)

Beschreibung: mfg_fill füllt das angegebene Fenster vom Punkt x, y (0, 0 ist linke obere Ecke des Bildschirms) in der Größe w (width) und h (height) mit der Farbe, die vorher mit Aufruf der Funktion mfg_setcolor(foreground, background) als Foreground-Colour festgelegt worden ist. Gefüllt wird im Bild (ram=1) oder im Overlay (ram=2).

Parameter: x,y: grafische Position auf dem Schirm (0 = links unten)
width, height: Größe des Rechtecks
ram=1: ins VRAM
ram=2: ins overlay

Beispiel: mfg_fill (x, y, w, h, ram)

2.4.5.7 mfg_c (x, y, width, height, bitmap, ram)

Beschreibung: mfg_c kopiert eine Bitmap, deren Länge durch acht teilbar sein muß, an eine beliebige Stelle auf den Bildschirm in den Bildspeicher oder ins Overlay.

Parameter: x,y: grafische Position auf dem Schirm (0 = links unten)
width, height: Größe des Rechtecks
bitmap: bitmap (xsize+7)/8 * ysize bytes
ram=1: to VRAM
ram=2: to overlay

Beispiel: mfg_c (x, y, width, height, bitmap, ram)

2.4.5.8 ms3_line (npoint, xyv, ram, plmode)

Beschreibung: ms3_line verbindet Punktepaare die im Vektor xyv (Pointer auf Anzahl npoint Punktepaare) beschrieben sind. ram unterscheidet zwischen Bild und Overlay, plmode legt fest ob die Vektoren untereinander verbunden sein sollen.

Parameter: xyv: vector of points, "npoint" points
ram=1/2 for write line into VRAM/overlay
plmode=0 -> connected polyline; 1->disjunct vectors

2.4.5.9 ms3_pblt (xs, ys, smask, xd, yd, dmask, width, height, ram)

Beschreibung: Kopiert im Bildschirmspeicher ein Rechteck von einer Adresse mit beliebigen Bit-Grenzen über eine Source-Maske in ein Rechteck mit einer Adresse in beliebigen Bit-Grenzen unter Benützung der Destination-Mask.

Diese Funktion wird benützt, um z.B.: Schrift in beliebiger Font-Größe oder einen Maus-Cursor zu bewegen.

Parameters:

xs, ys:	source (UL)
smask:	source bitplane mask
xd, yd:	dest (UL)
dmask:	dest bitplane mask
width, height:	size of rectangle
ram=1/2:	copy in VRAM / Overlay

2.4.5.10 vmfg_put2win (ptr, x, y, width, takepix, height, takeline, pitch, ram)

Beschreibung: vmfg_put2win schreibt Bildinformation ab einem Punkt x, y, (0, 0, ist linkes oberes Eck des Bildschirms) in ein Fenster der Breite w (width in pixel) und Höhe h (heigth in lines).

Das Bild kann verkleinert werden. Die Verkleinerung geschieht durch Weglassen von Pixel/ Zeilen, und kann unabhängig für x (takepix) oder y (takeline) eingestellt werden. (Maximum ist 6)

Die Bildinformation beginnt ab dem pointer ptr (long), wobei die gespeicherte Zeilenlänge = pitch ist. Die Breite „width“ muß so gewählt weden, daß sie ohne rest durch 2 x multx teilbar ist. Die Information wird in das Bild (ram=1) oder das Overlay (ram=2) geschrieben.

Parameter:

ptr:	UL corner of CPU image memory window to transfer
x, y:	UL corner of window in graphics display memory
width:	width of window [pixel] in dest device
takepix:	take each "takepix" pixel per line transfer (0->take all)
height:	height of window [lines] in dest device
takeline:	take each "takeline" line in dest device (0->take all)
pitch:	ptr memory address offset between lines
	example: mapping a 2560 pixel window to 640 display makes:
	width=640; takepix = 4; pitch=2560
destination:	ram=1:VRAM; 2:OVLY

2.4.5.11 vmfg_put4win (ptr, x, y, width, multx, height, multy, pitch, ram)

Beschreibung: vmfg_put4win schreibt Bildinformation ab einem Punkt x, y, (0, 0, ist linkes oberes Eck des Bildschirms) in ein Fenster der Breite w (width in pixel) und Höhe h (heigth in lines).
Jeder Pixel kann vergrößert dargestellt werden. Die Vergrößerung kann in x (multx) und y (multy) unabhängig eingestellt werden. Maximum ist 10.
Die Bildinformation beginnt ab dem pointer ptr (long), wobei die gespeicherte Zeilenlänge = pitch ist. Die Breite „width“ muß so gewählt weden, daß sie ohne rest durch 2 x multx teilbar ist. Die Information wird in das Bild (ram=1) oder das Overlay (ram=2) geschrieben.

Parameters:

ptr:	UL corner of CPU image memory window to transfer
x, y:	UL corner of window in graphics display memory
width:	width of window [pixel] in dest device
multx:	multiply each pixel "multx" times in X
height:	height of window [lines] in dest device
multy:	multiply each line "multy" times in Y
pitch:	ptr memory address offset between lines
	example: mapping a 512x512 pixel window 1024x1024 display makes: width=height=1024; multx=multy=2; pitch=512
destination:	ram=1:VRAM; 2:OVLY

2.4.5.12 linadrs3 ()

Beschreibung: linadrs3 ermöglicht den direkten linearen Zugriff auf den Bildspeicher der S3 Karte. Der Bildspeicher ist dann ab Adresse 2000000h (=32MB) in einer Länge von 1MB zugänglich. Die lineare Adresse zum Zugriff auf den VGA-Speicher erteilt das Betriebssystem . (DOS-Extender "Map Physical Memory")

linadrs3 ist die Voraussetzung für die Benützung der Funktion mvfg_dmawin.

2.4.5.13 mvfg_dmawin (x, y, width, height, pointer, pitch, color)

Beschreibung: mvfg_dmawin schreibt die Bildinformation direkt in den Bildspeicher der S3-VGA Karte. Es ist damit die mit Abstand schnellste Methode um Bilddaten zur Anzeige zu bringen und Vorraussetung für ein live Video Bild bei voller Auflösung. Der direkte Zugang zum VGA Bildspeicher muß zuvor mit der Funktion linadrs3 () geöffnet werden. Ein evtl. Overlay wird im dargestellten Bereich überschrieben.

Parameter:

x, y:	UL corner of window in graphics display memory
width:	width of window [pixel] in dest device
height:	height of window [lines] in dest device
ptr:	UL corner of CPU image memory window to transfer
pitch:	ptr memory address offset between lines
color=0	-> 8-Bit grey scale
color=1	-> 3/3/2 r/g/b color mode, ptr to other color planes is ptr+[pel_frm]
color=2	-> r/g/b 24 bit color mode ptr to other color planes is ptr+[pel_frm]
color=3	-> r=g=b grey mode

2.4.5.14 linends3 ()

Beschreibung: Beendet die lineare Addressierung des S3 Bildspeichers.

2.4.6 Sonstige Funktionen

2.4.6.1 mvfg_IntCallback (), mfg_IntCallback

Synopsis: void mvfg_IntCallback(* user_proc, segments);

Beschreibung: mvfg_IntCallback () übergibt der INSPECTA-Interrupt Service Routine die Adresse einer vom Benutzter geschriebenen Funktion (INSPECTA-Callback Funktion). Diese Funktion wird dann bei jedem Interrupt (alle 64 usec bei Zeileninterrupts oder alle 33 ms bei Bildinterrupts) aufgerufen.
Damit kann z.B.: eine Lichtschranke abgefragt werden und nachdem diese angesprochen hat, noch eine Anzahl von Zeilen gewartet werden, um dann eine mvfg_multiframe Funktion aufzurufen.

Der Parameter ‘segments’ ist für spätere Erweiterungen reserviert und hat aktuell immer den Wert 0.

Returnwert: --

Bemerkung: Diese Funktion steht unter DOSX und Windows95 zur Verfügung.
mvfg_IntCallback macht aus einer Benutzerfunktion eine Interrupt-Service Routine. Da eine Interrupt Service Routine auch auftreten kann, wenn die CPU im Betriebssystem arbeitet, dürfen in der Benutzerfunktion **keine** Betriebssystemaufrufe vorkommen. (zB.: kein printf).

2.4.6.2 mvfg_IntSource (), mfg_IntSource ()

Synopsis: void mvfg_IntSource (DWORD irqsource);

Beschreibung: Über ‘mvfg_IntSource()’ wird die Interruptquelle ausgewählt. Dies kann bei Verwendung der INSPECTA-Callback Funktion nützlich sein (siehe Fkt. ‘mvfg_IntCallback()’).

Parameter: ‘int_mode’ kann folgende Werte haben:

Bits:	0..7	= 0:	deactivate INTA
		> 0:	activate INTA
	8..9	= 0	
	10	= ISRC0 interrupt source 0	
	11	= ISRC1 interrupt source 1	
	12..32	= 0	

int_mode =	0x010	; activated by pixel clock/131072
	0x410	; activated by falling edge of VSYNC
	0x810	; activated by bit 0 of opto input rising edge
	0xC10	; activated by falling edge of HSYNC

Returnwerte: --

Bemerkung: Parameter int_mode Beschreibung für Inspecta, Hardware Revision >=4, IMP Nr.: >= IMP385, Software Rev. >=1.80

Beispiel: Aufruf der Benutzerdefinierten INSPECTA-Callback Funktion ca. alle 33 ms.

```
// Interruptquelle (VD) für den Aufruf der INSPECTA-Callback Funktion definieren.
mvfg_IntSource ( 0x0400 );

// INSPECTA-Callback Funktion bei INSPECTA anmelden.
mvfg_IntCallback( (DWORD) myCallBackFkt, 0 );
```

2.4.6.3 mfg_sync ()

mfg_sync () führt die Steuerungsfunktionen durch, die in den Austastlücken des Video-Signals gemacht werden müssen. So wird z.B.:
Bei Halbbildern bei jedem vertikalen Sync überprüft, ob das zweite Halbbild fertig ist.
Bei Farbsignalen nach jeweils zwei Halbbildern die Farbe gewechselt.
Für Bildsequenzen die Umschaltung der Kameras, die Einstellung des A/D Wandlers und die Belichtungssteuerung durchgeführt.
Wenn das Flag _fgtv_valid == 0 ist, ein Speicherblocktausch durchgeführt.
Wenn die mvfg_IntCallback Funktion aktiviert ist, die vom Aufrufer spezifizierte Funktion aufgerufen.

mfg_sync muß in jeder vertikalen oder horizontalen Austastlücke aufgerufen werden. Wenn die lokale Interrupt Service Routine *isr* aktiviert ist, ruft diese mfg_sync auf. Wenn die Anwendung eine eigene Service Routine installiert, muß von dieser aus mfg_sync aufgerufen werden.

mfg_sync beinflußt folgende counter & pointer:

DWORD int_cnt	; incrementiert bei jedem VD puls
DWORD frm_cnt	; incrementiert modulo frame_nr bei jedem Vollbild wenn rotierendes Schreiben gewählt ist, zeigt ansonsten immer das zum Schreiben gewählte Bild.
DWORD frame_counter	; incrementiert modulo frame_max bei jedem Vollbild wenn rotierendes Schreiben gewählt ist. (ab Ver. 1.99)
DWORD odd_flag	; =1 wenn gerade geschriebenes Halbbild ein ungerades war.
DWORD frm_beg	; offset innerhalb des frame memorys wo das zuletzt geschriebene Bild beginnt, nur wenn Bildspeicher <= 2x8MB ist
DWORD frm_begx	; offset innerhalb des frame memorys wo das zuletzt geschriebene Bild beginnt, auch wenn Bilspeicher größer 2x8MB ist (ab Ver. 1.99)
DWORD _fgtv_valid	; =1 wenn ein mfg_xchg durchgeführt wurde
DWORD integ_count	; Zähler für Integrationszeit, Anzahl Vollbilder
DWORD integration	; Gewünschter Integrationszähler
DWORD stop_start	; =1 wenn Start

2.4.6.4 mvfg_chkclk (), mfg_chkclk (),

Synopsis: DWORD mvfg_chkclk (void);

Beschreibung: Überprüft, ob für den gewählten Kameramodus ein Clocksignal vorhanden ist.

Returnwerte: 0: Clock vorhanden.
-2: Kein Clock vorhanden

Bemerkung: Diese Funktion wird vor dem Umschalten auf einen extern getakteten Kameramodus verwendet um zu überprüfen ob z.B.: eine passende Kamera am gewählten Eingang angesteckt ist.

Beispiel: --

2.4.6.5 mvfg_ppin (), mfg_ppin (),

Synopsis: DWORD mvfg_ppin (void);
Beschreibung: Einlesen des Zustands des digitalen I/O Ports der INSPECTA Karte.

Bits 0 .. 3 Zustand der 4 dig. Eingabeports.

Returnwerte: Status der I/O Ports.

Bemerkung: Bits 0..3 können nicht eingelesen werden, wenn Interrupt auf Bit 0 gewählt wurde. Siehe Funktionen mfg_int(), mvfg_IntSource(), mvfg_Snap()
Beispiel: --

2.4.6.6 mvfg_ppout (dout), mfg_ppout (dout)

Synopsis: void mvfg_ppout(DWORD value);
Beschreibung: Setzen der Bits am Ausgabeport des dig. I/O Ports der INSPECTA Karte.

Nur die Bits 0 .. 3, ensprechend den Ausgängen 0 .. 3 sind gültig.

Returnwerte: --

Beispiel: --

2.4.6.7 mfg_bmp (parameters)

Beschreibung: mfg_bmp speichert ganze Bilder oder Teile daraus auf Platte. Das Speicherformat ist ein unkomprimiertes *.BMP Format, daß mit allen WINDOWS Anwendumgen verarbeitet werden kann.

Parameter:

*file_name:	long pointer auf ASCIIZ String mit Filename	
* ptr:	long pointer auf Bildspeicher	
x:	x position des Schreibfensters, 0 = links oben	
y:	y position des Schreibfensters in Zeilen, 0 = erste Zeile	
width:	Breite des Schreibfensters in Pixel	
height:	Höhe des Schreibfensters in Zeilen	
pitch:	Zeilenlänge	
lut:	Look Up Table:	0 = 256 Grauwerte 2 = 24 Bit RGB Color, Color Auszüge in getrennten Bit-Planes
*buffer:	long pointer auf Kopierpuffer	
ploff:	Bildoffset, Abstand der Farbauszüge (in der Regel gleich Anzahl	Pixel pro Bild).

Rückgabewert: long 0 = o.K.
long x = File Error (create, write, disk full etc.)

Bemerkung: nur DOSX

2.4.6.8 m(v)fg_SetVideoClock (frequency)

Synopsis: void mvfg_ SetVideoClock(DWORD frequency);

Beschreibung: Setzt den Synthesizer zur Erzeugung des Video-Takts auf den angegebenen Wert (in Hz).

Returnwerte: --

Beispiel: --

Bemerkung: Ab Software Revision 1.62

2.4.6.9 mvfg_get_info(Z_MVFG_INFO * mvfg_info)

Synopsis: DWORD mvfg_get_info(Z_MVFG_INFO * mvfg_info)

Beschreibung: Die Funktion liefert Hard- und Software-Informationen des installierten INSPECTAS in der Struktur ‚mvfg_info' zurück..
Wird als Argument NULL übergeben, so wird lediglich die Seriennummer des installierten INSPECTAS als Funktionswert zurückgegeben.

Die Struktur Z_MVFG_INFO beinhaltet folgende Informationen:

Element	Beschreibung
driver_version_ms	Die oberen 32 Bit der Fileversion des verwendeten Device-Treibers
driver_version_ls	Die unteren 32 Bit der Fileversion des verwendeten Device-Treibers
dll_version_ms	Die oberen 32 Bit der Fileversion von ‚MVFGD32.DLL'
dll_version_ls	Die unteren 32 Bit der Fileversion von ‚MVFGD32.DLL'
hw_type	Typ des installierten INSPECTAS: 0 = undefiniert 1 = INSPECTA -1 2 = INSPECTA -2 3 = INSPECTA -3 4 = INSPECTA -4
hw_vendorID	Mikrotrons Vendor ID
hw_deviceID	INSPECTAS Device ID
hw_revID	INSPECTAS Revision ID
hw_snr	INSPECTA-4 Seriennummer
hw_imp	INSPECTAS Firmware Versionskennung

Returnwert:

!=	0	INSPECTA-4 Seriennummer
==	0	Fehler in Funktion aufgetreten

Beispiel:

```
Z_MVFG_INFO     mvfg_info;
.
mvfg_get_info( &mvfg_info );
```

Bemerkung: diese Funktion steht nur unter WinNT/2K ab Version 2.38 zur Verfügung

2.4.6.10 mvfg_get_info_ex(Z_MVFG_INFO_EX * mvfg_info_ex)

Synopsis: DWORD mvfg_get_info_ex(Z_MVFG_INFO_EX * mvfg_info _ex)

Beschreibung: Die Funktion liefert Hard- und Software-Informationen des installierten INSPECTAS in der Struktur ‚mvfg_info_ex' zurück..
Wird als Argument NULL übergeben, so wird lediglich die Seriennummer des installierten INSPECTAS als Funktionswert zurückgegeben.

Bei Aufruf der Funktion **mvfg_get_info_ex()** mit der Struktur Z_MVFG_INFO_EX als Parameter, müssen die Felder ‚**version**' und ‚**size**' mit gültigen Werten initialisiert sein!

Die Struktur Z_MVFG_INFO_EX erweitert die oben beschriebene Struktur Z_MVFG_INFO um einige zusätzliche Daten:

Element	Beschreibung
version	Version dieser Struktur (muss vor Aufruf der Funktion auf 2 gesetzt werden!)
size	Größe dieser Struktur in Bytes (muss vor Aufruf der Funktion gesetzt sein)
driver_version_ms	Die oberen 32 Bit der Fileversion des verwendeten Device-Treibers
driver_version_ls	Die unteren 32 Bit der Fileversion des verwendeten Device-Treibers
dll_version_ms	Die oberen 32 Bit der Fileversion von ‚MVFGD32.DLL'
dll_version_ls	Die unteren 32 Bit der Fileversion von ‚MVFGD32.DLL'
hw_type	Typ des installierten INSPECTAS: 0 = undefiniert 1 = INSPECTA -1 2 = INSPECTA -2 3 = INSPECTA -3 4 = INSPECTA -4
hw_sub_type	Typ des Hardware-Interfaces des INSPECTAS: 0 = Undefiniert 1 = ISA 2 = PCI 3 = PCI-X 4 = PCI-Express
hw_vendorID	Mikrotrons Vendor ID
hw_deviceID	INSPECTAS Device ID
hw_revID	INSPECTAS Revision ID

Element	Beschreibung
hw_snr	INSPECTA-4 Seriennummer
Element	Beschreibung
hw_imp	INSPECTAS Firmware Versionskennung
Hw_sub_system_id	INSPECTAS Sub System ID

Returnwert:

!=	0	INSPECTA-4 Seriennummer
==	-1	Fehler in Funktion aufgetreten

Beispiel:

```
Z_MVFG_INFO_EX  mvfg_info_ex;
.
mvfg_info_ex.version = 2;
mvfg_info_ex.size = sizeof(Z_MVFG_INFO_EX);
mvfg_get_info_ex( &mvfg_info_ex );
```

Bemerkung: diese Funktion steht nur unter WinNT/2K ab INSPECTA SetupVersion 4.34 zur Verfügung

2.4.7 Testfunktionen

Die in dieser Rubrik untergebrachten Bildausgabefunktionen mvga (parameter) und m13vga (parameter) sind hinsichtlich Geschwindigkeit und Auflösung nicht mehr zeitgemäß. Sie eignen sich aber sehr gut für Testzwecke, da sie mit jeder Standard-VGA arbeiten.

2.4.7.1 palette ()

Beschreibung: Diese Funktion setzt den VGA Adapter in mode 12h (640x480x16) und justiert die color palette zu einer linearen Grauwertdarstellung in 16 Stufen. Sie wird vor Benützung der Darstellungsfunktion mvga (....) aufgerufen.

2.4.7.2 mvga (alt_frame_start, frame_start, vga_start, granularity)

Beschreibung: mvga stellt einen Teil des Frame Buffers auf der Standard IBM kompatiblen VGA im VGA Modus 12h dar. Zusätzlich kann eine Differenzbildfunktion gewählt werden, in der jeder Grauwert eines Pixel mit dem gleichen Pixel des vorangegangenen Bildes subtrahiert werden. Die Differenz wird als Farbpunkt mit der Wertigkeit:
0 = schwarz
1 = blau
2 = grün
3 = cyan
4 = rot usw.
dargestellt wird.

Parameter:

alt_frame_start:	pointer to first pixel of image to compare.
frame_start:	pointer to first pixel of image to display.
vga_start:	pointer to first pixel on VGA screen.
granularity:	zoom down, ommit every second, third etc. pixel/line.

2.4.7.3 palette13

Beschreibung: palette13 schaltet den VGA Adapter in den VGA modus 13h.

2.4.7.4 m13vga (ptr, vga_ptr, granularity)

Beschreibung: m13vga schreibt Bilder in jede VGA Karte mit 320 x 200 x 8 Bit Auflösung. Die Ausgabegeschwindigkeit ist wegen der single plane Anordnung (8 Bit pro Pixel) und der geringen Datenmenge sehr schnell. Zur Initialisierung muß vorher palette928 (0) und palette13 () aufgerufen werden.

Parameter:

ptr:	pointer to first pixel of image to display.
vga_ptr:	pointer to first pixel on VGA screen.
granularity:	zoom down, ommit every second, third etc. pixel/line.