User manual

FALCON
ActiveX
Software Interface

FALCON Frame Grabber Family

Version 1.1

Status: January 2008

Dimbacher Strasse 6
D-74182 Obersulm
Fax: +49/(0)7134/96196-99
eMail: sales@ids-imaging.de

Preface

IDS has compiled this user manual with the greatest possible care. However, we assume no liability for the content, completeness or quality of the information in this manual. The content of this manual is updated and adapted to reflect the current status of the software. We furthermore do not guarantee that this product will function without errors, even if the stated specifications and recommended hardware configuration are adhered to.

Under no circumstances can we guarantee that a particular objective can be achieved with the purchase of this product.

Insofar as permitted under statutory regulations, we assume no liability for direct damage, indirect damage or damages suffered by third parties resulting from the purchase of this product. In no event shall any liability exceed the purchase price of the product.

Status: January 2008

Safety Notes
We point out that the content of this manual is not a part of an earlier declaration/agreement. All guarantees are based on your license agreement, when the system was purchased.
If you need further information or if you have special problems which are not mentioned in this manual, you can contact your installer or the address listed below.
The installation and maintenance must be done by qualified persons.
The correct and secure function of this system is based on careful transport, correct storing, installation and maintenance.

Registered trade mark
IBM PC is a registered trademark of the International Business Machines Corporation and Windows is a trademark of the Microsoft Corporation. All other products or firms mentioned in this manual serve only for the purpose of identification or description and can be registered trade marks or entered registered trade marks of the respective owners.

Contacting us
Visit our web site where you will find all the latest drivers and information about our software and hardware products as well as our partners and distributors.

Internet: http://www.ids-imaging.com

Address: IDS Imaging Development Systems GmbH
Dimbacher Strasse 6
D-74182 Obersulm
Germany
Fax: +49 (0) 77134/96196-99
Email: Sales: sales@ids-imageing.com
Support: support@ids-imaging.com

Contents

1 General

Scope of Delivery
Unless agreed otherwise, the scope of delivery of the ActiveX Control includes the following components:

- Setup program with Active X Control for Windows NT 4.0, Windows 2000/XP and Windows Vista 32-Bit, plus example programs.

System Requirements

- Windows NT4.0, Windows 2000/XP or Windows Vista 32-Bit
- Correctly installed FALCON or EAGLE frame grabber with driver v. 2.10 or higher. Please observe the instructions in the frame grabber manual when installing the frame grabber.
- PCI/AGP VGA card with DirectDraw support (not mandatory, but achieves the best results)

2 Installation

2.1 Windows NT 4.0, Windows 2000/XP, Windows Vista 32-Bit

To install the software, run the program SETUP.EXE. A menu will guide you through the self-explanatory set-up process. The installation program will copy the following files:

falcon.ocx to \<windir>\system32
mfc42.dll to \<windir>\system32 (if this file does not yet exist)
msvcrt.dll to \<windir>\system32 (if this file does not yet exist)

In addition, the following directory structure is created:

<Install>\IDS\Falcon\ActiveX
\Develop\Source\Borland C++ Builder 1.0
\Develop\Source\VB6
\Develop\Source\VC
\Help
\Samples
\Tools

(*<windir>* refers to the Windows root directory.
<Install> is the directory you specified during the installation)

3 FALCON/EAGLE ActiveX Function List

3.1 Initialising and terminating

Init	Initialise the hardware
Exit	Shut down the card and clear the image buffer

3.2 Image capture and memory management

StopLiveVideo	Freeze image
CaptureVideo	Capture live video
FreezeVideo	Capture an image and write it to a target address, Snap
HasVideoStarted	Has image acquisition started?
IsVideoFinish	Has image acquisition ended?
InquireImageMem	Returns properties of the image buffer
GetBitmapHandle	Get bitmap handle
GetImageMem	Get a pointer to the image buffer

3.3 Selecting operating modes and reading out settings

BoardStatus	Returns event counters and counter values
SetVideoInput	Select video input
SetBrightness	Set brightness of the image
SetContrast	Set contrast
SetSaturation	Set colour saturation
SetHue	Set hue
SetVideoMode	Select video standard
SetDisplayMode	Select image display mode
SetAGC	Activate/deactivate automatic gain control
SetGamma	Activate/deactivate Gamma correction
SetSyncLevel	Set sync level for critical video sources
SetColorMode	Select colour mode
SetScaler	Activate/deactivate the scaler; set scale percentage
SetImageSize	Set image size
SetImagePos	Set image position within the image window
SetCaptureMode	Select a capture mode
SetErrorReport	Activate/deactivate error reporting
SetHorFilter	Set horizontal interpolation filter
SetVertFilter	Set vertical interpolation filter
GetError	Get error number
GetErrorString	Get error message
GetCurrentField	Returns the currently active field (odd or even)
GetOsVersion	Get operating system
SetVideoSize	Specify the image area to be digitised

3.4 Reading and writing to the EEPROM

WriteEEPROM	Write own data to the EEPROM
ReadEEPROM	Read own data from the EEPROM
GetBoardType	Get the grabber type (FALCON or EAGLE)
GetBoardVersion	Get the hardware version of the grabber
GetSerialNumber	Get the serial number of the grabber

3.5 Saving and loading images

SaveImage	Save a video image as a BMP file
LoadImage	Load a BMP image

3.6 Image output

UpdateDisplay	Update screen when using DirectDraw
ShowColorBars	Display colour bars (hardware)

4 Description of the functions

The functions and parameters described in this chapter are provided for integrating the FALCON/EAGLE Active X Control into your own programs. The functions are listed in alphabetical order and structured as follows:

<Name of the function>

Syntax:

Function prototype

Description:

Describes the function, cross-referencing related functions.

Transfer parameter:

Describes the parameters required by the function and their value ranges.

Return value:

Describes return values and their value ranges. If a function returns the value IS_NO_SUCCESS (-1), the error can be retrieved using the functions GetError() or GetErrorString(). If a function returns IS_SUCCESS(0), it was executed without encountering any errors..

4.1 BoardStatus

Syntax:

LONG BoardStatus (LONG Info, LONG Value)

Description:

BoardStatus() can be used to set and query the fifo overrun counter.

Transfer parameter:

nInfo	IS_FIFO_OVR_CNT	= 1
	IS_GET_STATUS	= 8000 Hex
lValue	Set the counter given in nInfo to the corresponding value (e.g. reset counter: lValue = 0)	

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.2 CaptureVideo

Syntax:

LONG CaptureVideo (LONG Wait)

Description:

CaptureVideo() digitises video images in real time and transfers the images to an allocated image buffer or, under DirectDraw, to the graphics card. The image capture parameters set with the *SetCaptureMode()* function are taken into account.

Transfer parameter:

Wait

1	IS_WAIT	Function synchronises image capture to the next frame or field VSYNC and only returns then
0	IS_DONT_WAIT	Function synchronises image capture to the next frame or field VSYNC but returns immediately
	4000 Hex IS_FORCE_VIDEO_START	Starts digitisation immediately (unsynchronised) and returns immediately

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.3 DisableDDOverlay

Syntax:

LONG DisableDDOverlay ()

Description:

DisableDDOverlay() deactivates the overlay mode in DirectDraw back buffer mode and frees up the memory used by the overlay. The overlay data are discarded.

Transfer parameter:

<none>

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.4 EnableDDOverlay

Syntax:

LONG EnableDDOverlay ()

Description:

EnableDDOverlay() activates live overlay mode in DirectDraw back buffer mode. Three non-visible image buffers are used in overlay mode: back buffer, overlay buffer and mix buffer. The video image is digitised to the back buffer. The image data can be written to the overlay buffer. The back buffer and the overlay buffer are then jointly written to the mix buffer. The overlay data are superimposed on the video image. The mix buffer is then copied to the visible area of the VGA card. The three buffers each have the following size: Video_X * Video_Y * colour depth (in bytes per pixel) The driver attempts to allocate the buffers directly within the VGA card in order to take advantage of the VAG card's high-speed image transfer capabilities when mixing the three buffers. If the buffers cannot be allocated within the VGA card, the system memory is used. Image transfer from the system memory is however slower or may be entirely impossible under certain circumstances (depending on the graphics card). The overlay is not displayed automatically. It must first be made visible using *ShowDDOverlay()*. The overlay uses the colour black as a colour key; therefore overlay graphics must not contain the colour black (use an almost black colour as a workaround, e.g. dark blue).

Transfer parameter:

<none>

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.5 Exit

Syntax:

LONG Exit ()

Description:

Exit() shuts down the ActiveX Control, closes the frame grabber and frees up the memory used.

Transfer parameter:

<none>

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.6 FreezeVideo

Syntax:

LONG FreezeVideo (LONG Wait)

Description:

FreezeVideo() digitises an image and stores it in the image buffer. In DirectDraw mode the image is digitised to the DirectDraw buffer (directly to the VGA card or a back buffer).

Transfer parameter:

Wait

1	IS_WAIT	Function waits until the image has been captured
0	IS_DONT_WAIT	Function returns immediately
10 < Wait < 32768		Wait time in 10ms steps (Wait = 100 Þ wait 1 sec) The maximum wait time is 328.68 seconds (approximately five minutes and 20 seconds). For 1 < Wait < 10, Wait = 10 is set.

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.7 GetBitmapHandle

Syntax:

OLE_HANDLE GetBitmapHandle ()

Description:

GetBitmapHandle() fetches the handle to a bitmap that corresponds to the image supplied by the frame grabber.

Transfer parameter:

<none>

Return value:

Handle to a bitmap

4.8 GetBoardType

Syntax:

BSTR GetBoardType ()

Description:

If both FALCON and EAGLE are simultaneously integrated, it is absolutely necessary to distinguish between the respective board types. *GetBoardType()* returns the type of board (FALCON / EAGLE) as a result.

Transfer parameter:

<none>

Return value:

“FALCON” or “EAGLE” as a string

4.9 GetBoardVersion

Syntax:

BSTR GetBoardVersion ()

Description:

GetBoardVersion() reads out the EEPROM of the frame grabber and determines the version number of the hardware.

Transfer parameter:

<none>

Return value:

e.g. “V2.00” as a string

4.10 GetCurrentField

Syntax:

LONG GetCurrentField ()

Description:

GetCurrentField() indicates which field is currently active (odd or even)

Transfer parameter:

<none>

Return value:

0 = even, 1 = odd

4.11 GetDC

Syntax:

OLE_HANDLE GetDC ()

Description:

GetDC() returns the device context handle of the overlay buffer. This handle enables access to the overlay via the Windows GDI functions. All Windows graphics commands such as the Line, Circle, Rectangle, TextOut,… functions are thus available. The device context handle must be released as soon as possible using the *ReleaseDC()* function. The overlay buffer on screen will **NOT** be updated within a *GetDC – Release DC* block.

Transfer parameter:

<none>

Return value:

Device context handle or NULL

4.12 GetError

Syntax:

LONG GetError ()

Description:

GetError() gets the last encountered error and returns error code and error message. The last error message is not deleted, but is overwritten by new errors as they occur.

Transfer parameter:

<none>

Return value:

Number (error code) of the error encountered

4.13 GetErrorString

Syntax:

BSTR GetErrorString ()

Description:

GetErrorString() gets the last encountered error and returns error code and error message. The last error message is not deleted, but is overwritten by new errors as they occur.

Transfer parameter:

<none>

Return value:

A string describing the error.

4.14 GetImageMem

Syntax:

VARIANT GetImageMem ()

Description:

GetImageMem() returns a pointer to the image buffer wrapped in a VARIANT data type. The pbVal element of the union contains the pointer to the image buffer.

Transfer parameter:

<none>

Return value:

VARIANT data type, the pbVal element of which contains the pointer to the image buffer.

Example:

```
VARIANT var = Falcon.GetImageMem ();
unsigned char* pucImageMem = var.pbVal;
```

4.15 GetOsVersion

Syntax:

LONG GetOsVersion ()

Description:

GetOsVersion() returns the type of operating system in use at runtime.

Transfer parameter:

<none>

Return value:

IS_OS_WINNT40 = 2, IS_OS_WIN2000 = 4, IS_OS_WINXP = 5, IS_OS_WINVISTA = 7

4.16 GetSerialNumber

Syntax:

BSTR GetSerialNumber ()

Description:

GetSerialNumber() reads the EEPROM of the frame grabber and returns the grabber's serial number.

Transfer parameter:

<none>

Return value:

e.g. "*12345*" as a string

4.17 HasVideoStarted

Syntax:

BOOL HasVideoStarted ()

Description:

HasVideoStarted() can be used to check whether image digitisation has started. This function is useful in combination with *FreezeVideo()* and the IS_DONT_WAIT parameter.

Transfer parameter:

<none>

Return value:

TRUE(1) if image acquisition has started, otherwise FALSE (0)

4.18 HideDDOverlay

Syntax:

LONG HideDDOverlay ()

Description:

HideDDOverlay() hides the overlay in DirectDraw back buffer mode. Only the image buffer is then displayed, resulting in a faster refresh rate than when the overlay is shown. The overlay data are not lost by hiding the overlay.

Transfer parameter:

<none>

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.19 Init

Syntax:

LONG Init ()

Description:

Init() opens the driver and connects to the hardware. The parameters set in the Properties dialogue are transmitted to the hardware during initialisation. These parameters can also be changed at runtime before *Init()* is called. For details see the description of parameters at the end of this manual. All other functions can only be executed after an *Init()*. If the „InitAtStartup“ property is set, an *Init()* is executed immediately when the ActiveX Control is started up. If „InitAtStartup“ is not set, *Init()* must be called explicitly within the program.

Transfer parameter:

<none>

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.20 InquireImageMem

Syntax:

LONG InquireImageMem (long* pnX, long* pnY, long* pnBits, long* pnPitch)

Description:

is_InquireImageMem() reads out the properties of an allocated image buffer.

Transfer parameter:

pnX	Contains the width with which the image buffer was created
pnY	Contains the height with which the image buffer was created
pnBits	Contains the bit width with which the image buffer was created
pnPitch	Contains the line pitch of the image buffer

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.21 IsVideoFinish

Syntax:

BOOL IsVideoFinish ()

Description:

IsVideoFinish() can be used to check whether an image has been completely written to the image buffer. This function is useful in combination with *FreezeVideo()* and the IS_DONT_WAIT parameter.

Transfer parameter:

<none>

Return value:

TRUE(1) if image acquisition has ended, otherwise FALSE (0)

4.22 LoadImage

Syntax:

LONG LoadImage (BSTR Filename)

Description:

Loads an image from a file. The image must be in BMP format. The image is loaded into the image buffer or, under DirectDraw, into the corresponding DirectDraw buffer. Please note that in DirectDraw primary surface mode the image will be loaded at a size no larger than the output window. If the image to be loaded is larger, the overhanging areas will be cut off.

The file name may contain both absolute and relative file paths. Use *SaveImage()* to save images as BMP files.

Transfer parameter:

Filename	Name of the BMP image file, NULL or empty string -> dialogue box for entering a file path appears

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.23 OvlSurfaceOffWhileMove

Syntax:

LONG OvlSurfaceOffWhileMove (BOOL boMode)

Description:

OvlSurfaceOffWhileMove() deactivates the overlay surface in overlay surface mode while the output window is moved or resized, or when other applications are activated. While the overlay surface is deactivated, the standard background colour of the window is shown.

Transfer parameter:

boMode	TRUE	Deactivate overlay while window is moved
	FALSE	Overlay always active (default)

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.24 PropertyDialog

Syntax:

LONG PropertyDialog ()

Description:

PropertyDialog() calls up the property dialogue. This dialogue allows changes to the frame grabber settings to be made at runtime.

Transfer parameter:

<none>

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.25 ReadEeprom

Syntax:

BSTR ReadEeprom ()

Description:

A rewriteable EEPROM is located on the frame grabber as a small storage unit. In addition to the information stored permanently on the card, 64 bytes of own data can be written to the EEPROM. The contents of this 64-byte block can be read out with the *ReadEEPROM()* function. See also *WriteEEPROM().*

Transfer parameter:

<none>

Return value:

Content of the EEPROM as a string

4.26 ReleaseDC

Syntax:

LONG ReleaseDC (OLE_HANDLE hDC)

Description:

ReleaseDC() releases the device context handle of the overlay buffer in DirectDraw back buffer mode. After the handle is released, the overlay buffer is updated on the screen (if the overlay is displayed with *ShowDDOverlay()*).

Transfer arameter:

hDC	device context handle from *GetDC()*

Return value:

IS_SUCCESS oder IS_NO_SUCCESS

4.27 SaveImage

Syntax:

LONG SaveImage (BSTR Filename)

Description:

Saves an image to a file in bitmap format (*.BMP). The images are read out from the active image buffer. In the DirectDraw modes the image data are read directly from the corresponding DirectDraw buffer. Please note that in DirectDraw primary surface mode the image will be saved as it is currently displayed on screen. It is only saved at the size of the image output window at most, irrespective of the image size set with *SetImageSize()*. If other windows overlap the image output window entirely or partially, the content of the overlapping windows is also saved in the bitmap file!
The bitmap is stored at the same colour depth (8, 15, 16, 24 or 32-bit) that is set as the current colour mode for DirectDraw output modes. Some image editing programs do not support 15-bit, 16-bit or 32-bit bitmaps and will therefore not be able to read images saved in these modes!
The file name may contain both absolute and relative file paths. BMP images are read with *LoadImage()*.
Overlay data are not saved!

Transfer parameter:

File	Name of the BMP image file NULL or empty string ->“Save as“ dialogue box opens

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.28 SetAGC

Syntax:

LONG SetAGC (LONG Mode)

Description:

SetAGC() activates or deactivates AGC (Automatic Gain Control).

Transfer parameter:

Mode

IS_SET_AGC_OFF	Deactivate AGC
IS_SET_AGC_ON	Activate AGC
IS_GET_AGC_MODE	Read out current setting

Return value:

Current setting in combination with IS_GET_AGC_MODE, otherwise IS_SUCCESS, IS_NO_SUCCESS

4.29 SetBrightness

Syntax:

LONG SetBrightness (LONG Brightness)

Description:

SetBrightness() adjusts the brightness of the image. The *Brightness* parameter can take a value between 0 and 255. Brightness is adjusted by changing the luminance value by means of an offset set in *Brightness*. The default *Brightness* value is 128 (IS_DEFAULT_BRIGHTNESS).

Transfer parameter:

Brightness	Brightness in the range from 0..255
IS_GET_BRIGHTNESS	Read out current setting

Return value:

Current setting in combination with IS_GET_BRIGHTNESS, otherwise IS_SUCCESS, IS_NO_SUCCESS

4.30 SetCaptureMode

Syntax:

LONG SetCaptureMode (LONG Mode)

Description:

SetCaptureMode() sets the desired digitisation mode.
It supports capturing only even video images (even fields), only odd video images (odd fields) or entire interlaced video images (frames). If you need to capture both fields but at the same time want to display them underneath each other, you can achieve this with the additional parameter IS_SET_CM_NONINTERLACED. The parameter IS_SET_CM_FRAME_STEREO activates stereo capture mode. In this stereo mode two synchronised cameras must be connected to the two composite video inputs (inputs 1 and 2). During image acquisition the grabber then switches to the respective other camera in the vertical blanking interval (VSYNC pulse). The result is that e.g. the first field is always captured by camera 1 and the second field by camera 2. In interlaced display a 3D-image can be viewed with 3D shutter glasses.

Transfer parameter:

Mode	Capture odd fields
IS_SET_CM_ODD	
IS_SET_CM_EVEN	Capture even fields
IS_SET_CM_FRAME	Capture video images interlaced
IS_SET_CM_NONINTERLACED	Capture video images non-interlaced
IS_SET_CM_FRAME_STEREO	Capture stereo images interlaced
IS_GET_CAPTURE_MODE	Read out current setting

Return value:

Current setting in combination with IS_GET_CAPTURE_MODE, otherwise IS_SUCCESS, IS_NO_SUCCESS

4.31 SetColorMode

Syntax:

LONG SetColorMode (LONG Mode)

Description:

SetColorMode() sets the desired colour mode in which the image data are saved and displayed in the VGA card. In the former case, depending on the colour mode used it is important that the allocated image buffer is large enough. A 24-bit colour image requires three times the memory size of an 8-bit monochrome image. When accessing image data, it is important to understand how memory is organised in the various colour modes. Incorrect interpretation of the memory contents leads to incorrect results. When transferring directly to the image buffer of the VGA card, it must be ensured that the display settings correspond to the colour mode settings of the grabber. Otherwise the images may be displayed off-colour or become unrecognisable. Please also note: The greater the pixel depth of the transferred images, the greater the load on the PCI bus. It is therefore recommended to select greyscale mode for monochrome images and to also set the display to 256 colours.

Transfer parameter:

Mode

IS_SET_CM_RGB32	32-bit true colour mode; R-G-B dummy
IS_SET_CM_RGB24	24-bit true colour mode; R-G-B
IS_SET_CM_RGB16	Hi-Color mode; 5 R - 6 G - 5 B
IS_SET_CM_RGB15	Hi-Color mode; 5 R -5 G - 5 B
IS_SET_CM_YUV422	YUV4:2:2 (not displayed in VGA card)
IS_SET_CM_YUV411	YUV4:1:1 (not displayed in VGA card)
IS_SET_CM_Y8	8-bit monochrome images
IS_SET_CM_RGB8	8-bit colour images; 2 R - 3 G - 2 B
IS_GET_COLOR_MODE	Read out current setting

Return value:

Current setting in combination with IS_GET_COLOR_MODE, otherwise IS_SUCCESS, IS_NO_SUCCESS

4.32 SetContrast

Syntax:

LONG SetContrast (LONG Contrast)

Description:

SetContrast() adjusts the contrast of the image (increases luminance) in the range between 0% and 200%.

Transfer parameter:

Contrast	Contrast; values between 0..511 or IS_GET_CONTRAST (8000 Hex)

Return value:

Current setting in combination with IS_GET_CONTRAST, otherwise IS_SUCCESS, IS_NO_SUCCESS

4.33 SetDDUpdateTime

Syntax:

LONG SetDDUpdateTime (LONG ms)

Description:

is_SetDDUpdateTime() sets the timer interval for the video image update cycle in DirectDraw mode back buffer mode. Permitted values are 20ms to 2000ms (20ms is unrealistic).

Transfer parameter:

ms time in milliseconds

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.34 SetDisplayMode

Syntax:

LONG SetDisplayMode (LONG Mode)

Description:

SetDisplayMode() determines how the images are displayed on the screen. DirectDraw overlay surface mode was introduced to enable real live video plus overlay. Whether this mode is available depends on your VGA card. Only certain VGA controllers support this mode. Currently known models are: S3 Virge VX (to a limited degree), S3 Virge DX/GX/GX2 and the 3D RagePro chip. This overlay mode can be used with these VGA chips in 8-bit, 15-bit and 16-bit VGA mode. True Color (24-bit) is not supported. In 8-bit VGA mode the colour mode must be set to RGB15!
The memory size of the VGA card should be 4MB, as the overlay mode requires memory space up to the size of the current VGA resolution. Example: VGA with 1024x768x16 = 1.5 MB -> overlay buffer up to 1.5 MB
For a VGA resolution of 640x480, image size must be limited to 640x480 pixels before activating DirectDraw mode. The function used for this purpose is *SetImageSize().*

Transfer parameter:

Mode	Camera handle
Basic mode	
IS_SET_DM_DIB	Capture image in image memory
IS_SET_DM_DIRECTDRAW	DirectDraw mode (back buffer mode)
DirectDraw back buffer extension	
IS_SET_DM_ALLOW_SYSMEM	DirectDraw buffer can be created on the PC if VGA memory is insufficient
DirectDraw primary surface extension	
IS_SET_DM_ALLOW_PRIMARY	Primary Surface Mode (no overlay possible)
DirectDraw overlay surface extension	
IS_SET_DM_ALLOW_OVERLAY	Overlay surface mode
IS_SET_DM_ALLOW_SCALING	Real time scaling in overlay surface mode
IS_SET_DM_ALLOW_FIELDSKIP	Field display for ysize £ PAL/2 (288)
Return mode	
IS_GET_DISPLAY_MODE	Return the current settings

Return value:

Current setting in combination with IS_GET_DISPLAY_MODE, otherwise IS_SUCCESS, IS_NO_SUCCESS

Example:

SetDisplayMode (Mode);
bitmap mode (digitized in system memory):
Mode = IS_SET_DM_DIB

DirectDraw back buffer mode (overlay possible. See *is_EnableDDOverlay):*
Mode = IS_SET_DM_DIRECTDRAW

or in order to enable swapping to system memory:
Mode = IS_SET_DM_DIRECTDRAW | IS_SET_DM_ALLOW_SYSMEM

DirectDraw primary surface mode (no overlay possible):
Mode = IS_SET_DM_DIRECTDRAW | IS_SET_DM_ALLOW_PRIMARY

DirectDraw overlay surface mode (best live overlay):
Mode = IS_SET_DM_DIRECTDRAW | IS_SET_DM_ALLOW_OVERLAY

or in order to enable automatic scaling to window size:
Mode = IS_SET_DM_DIRECTDRAW | IS_SET_DM_ALLOW_OVERLAY | IS_SET_DM_ALLOW_SCALING

or in order to prevent the interlaced effect at y<288:
Mode = IS_SET_DM_DIRECTDRAW | IS_SET_DM_ALLOW_OVERLAY | IS_SET_DM_ALLOW_SCALING | IS_SET_DM_ALLOW_FIELDSKIP

Other combinations for *Mode* are invalid!

4.35 SetErrorReport

Syntax:

LONG SetErrorReport (LONG Mode)

Description:

Activates or deactivates error reporting. If error reporting is active, any errors that occur are automatically displayed in a dialogue box. If the dialogue box is closed with *„Cancel”*, error reporting is also deactivated. If error reporting is not active, errors can be queried using the functions *GetError()* and *GetErrorString()*.

Transfer parameter:

Mode	0 IS_DISABLE_ERR_REP 1 IS_ENABLE_ERR_REP

Return value

Current setting in combination with IS_GET_ERR_REP_MODE, otherwise IS_SUCCESS, IS_NO_SUCCESS

4.36 SetExternalTrigger (EAGLE only)

Syntax:

LONG SetExternalTrigger (LONG Mode)

Description:

SetExternalTrigger() activates the trigger input. The edge on which an „event“ is to be triggered is passed in the function call:

Activity on High-Low edge (TTL)	IS_SET_TRIG_HI_LO
Activity on Low-High edge (TTL)	IS_SET_TRIG_LO_HI

Under Windows NT 4.0, an EVENT is signalled with every trigger occurrence. This must however first be activated by means of the function is_Enable_Event (parameter *IS_SET_EVENT_EXTTRIG*). If only one EVENT is to be triggered under Windows NT 4.0, *IS_SET_TRIG_HI_LO* or *IS_SET_TRIG-LO_HI* must be linked with *IS_SET_TRIG_EVENT_ONLY* by an OR operator.

Otherwise, two trigger modes are distinguished:
1. If the trigger input is set to active, the following field or image (next following VSYNC) is captured when a trigger signal is received after calling the *FreezeVideo()* function. Whether an image or a field is captured depends on the settings in *SetCaptureMode()*.
2. If the trigger input is set to active, the current digitisation is stopped if a trigger signal is received after calling the *StopLiveVideo()* function. The field or image last digitised is captured. Whether field processing is to take place here also depends on the settings of the *SetCaptureMode()* function. Previous live image capture (*CaptureVideo()*) is a prerequisite for this procedure.
3. If the trigger input is set to active, digitisation is started when a trigger signal is received after calling the *CaptureVideo()* function.

The modes are distinguished by calling the corresponding functions (*FreezeVideo(), StopLiveVideo(), CaptureVideo()*). For this purpose the WAIT parameter was introduced for these functions (timeout criterion).

Transfer parameter:

Mode

IS_SET_TRIG_OFF	= 0
IS_SET_TRIG_HI_LO	= 1
IS_SET_TRIG_LO_HI	= 2
IS_SET_TRIG_EVENT_ONLY	= 4 (only WinNT 4.0)
IS_GET_EXTERNALTRIGGER	= 8000 Hex

Return value:

IS_SUCCESS, IS_NO_SUCCESS or current setting in combination with IS_GET_EXTERNALTRIGGER.

4.37 SetFlashStrobe (EAGLE only)

Syntax:

LONG SetFlashStrobe (LONG Field, LONG Line)

Description:

SetFlashStrobe() activates the flash strobe. The field for which a strobe is to be triggered can be specified in the *Field* parameter. If a strobe is to be triggered on both fields, both parameters must be joined by an OR operator (EVEN + ODD). By default the strobe is low active. The strobe signal can be inverted with *IS_SET_FLASH_HI_ACTIVE* for nField. The Line parameter can then be used to set the line number at which the strobe is to be triggered. Strobe duration is approximately 64 microseconds.

Parameters:

Field	
IS_SET_FLASH_OFF	= 0
IS_SET_FLASH_ODD	= 1000 Hex
IS_SET_FLASH_EVEN	= 2000 Hex
IS_SET_FLASH_HI_ACTIVE	= 4000 Hex
IS_GET_FLASHSTROBE_FIELD	= 8000 Hex
IS_GET_FLASHSTROBE_LINE	= 8001 Hex
Line	Line at which the strobe impulse is triggered

Return value:

IS_SUCCESS, IS_NO_SUCCESS,
the current mode in connection with IS_GET_FLASHSTROBE_FIELD or
the current line in connection with IS_GET_FLASHSTROBE_LINE

4.38 SetGamma

Syntax:

LONG SetGamma (LONG Gamma)

Description:

SetGamma() activates or deactivates Gamma correction. Gamma correction is performed on colour signals to adjust the colour values to the monitor. In Y8 monochrome mode Gamma correction has no effect.

Transfer parameter:

Gamma	
IS_SET_GAMMA_OFF	Deactivate Gamma correction
IS_SET_GAMMA_ON	Activate Gamma correction
IS_GET_GAMMA_MODE	Read out current setting

Return value:

Current setting in combination with IS_GET_GAMMA_MODE, otherwise IS_SUCCESS, IS_NO_SUCCESS

4.39 SetHorFilter

Syntax:

LONG SetHorFilter (LONG Mode)

Description:

SetHorFilter() activates the horizontal interpolation filters. This function is only useful if image scaling is activated. Select a filter level by linking it with IS_ENABLE_HOR_FILTER by means of an OR operator.

Transfer parameter:

Mode

IS_DISABLE_HOR_FILTER	Deactivate the filter
IS_ENABLE_HOR_FILTER	Activate the filter
IS_HOR_FILTER_STEP1	Filter level 1 (low filtering)
IS_HOR_FILTER_STEP2	Filter level 2
IS_HOR_FILTER_STEP3	Filter level 3 (strong filtering)
IS_HOR_FILTER_MODE	Read out current setting
IS_HOR_FILTER_STEP	Read out current setting

Return value:

Current setting in combination with IS_GET_HOR_FILTER_MODE or IS_GET_HOR_FILTER_STEP, otherwise IS_SUCCESS, IS_NO_SUCCESS

4.40 SetHue

Syntax:

LONG SetHue (LONG Hue)

Description:

The *SetHue()* function adjusts the image hue. This function is useful in combination with an NTSC colour camera. It has no effect on monochrome images.

Transfer parameter:

Hue	value range 0..255;
IS_GET_HUE	

Return value:

Current setting in combination with IS_GET_HUE, otherwise IS_SUCCESS, IS_NO_SUCCESS

4.41 SetImagePos

Syntax:

LONG SetImagePos (LONG x, LONG y)

Description:

SetImagePos() determines the starting point for digitisation. In combination with the *SetImageSize()* function, an area can be cut out of the full video image. The parameters x and y represent an offset from the upper left corner of the image.

Transfer parameter:

x IS_GET_IMAGE_POS_X	Horizontal position; value range: 1..767
y IS_GET_IMAGE_POS_Y	Vertical position; value range: 1..575

Return value:

Current setting in combination with IS_GET_IMAGE_POS_X and IS_GET_IMAGE_POS_Y as parameters for x, otherwise IS_SUCCESS, IS_NO_SUCCESS

4.42 SetImageSize

Syntax:

LONG SetImageSize (LONG width, LONG height)

Description:

SetImageSize() determines the image size in combination with the settings made with *SetImagePos()*.

Transfer parameter:

width IS_GET_IMAGE_SIZE_X	Image width; value range 0..768
height IS_GET_IMAGE_SIZE_Y	Image height; value range 0..576

Return value:

In combination with IS_GET_IMAGE_SIZE_X and IS_GET_IMAGE_SIZE_Y the current settings are read, otherwise IS_SUCCESS or IS_NO_SUCCESS.

4.43 SetIO (EAGLE only)

Syntax:

LONG SetIO (LONG io)

Description:

SetIO() sets the digital output or reads the digital input.

Transfer parameter:

nIO 0	Delete output 1 Set output
IS_GET_IO	Read input

Return value:

IS_SUCCESS, IS_NO_SUCCESS or for nIO = IS_GET_IO the value of the digital input

4.44 SetKeyColor

Syntax:

LONG SetKeyColor (LONG r, LONG g, LONG b)

Description:

SetKeyColor() defines the key colour for DirectDraw overlay surface mode.

Transfer parameter:

hf	Camera handle
r	Red channel of keying colour (0...255).
IS_GET_KC_RED	Retrieval of the R channel
IS_GET_KC_GREEEN	Retrieval of the G channel
IS_GET_KC_BLUE	Retrieval of the B channel
IS_GET_KC_RGB	Retrieval of the RGB
IS_GET_KC_INDEX	Returns the colour index
g	Green channel of keying colour (0...255).
b	Blue channel of keying colour (0...255).

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.45 SetKeyOffset

Syntax:

LONG SetKeyOffset (LONG xOffset, LONG yOffset)

Description:

SetKeyOffset() moves the display position of the image by +/- 1 pixel in x and y direction in DirectDraw overlay surface mode. Normally this should not be necessary, but it depends on the VGA card and driver used.

Transfer parameter:

xOffset	X offset, permitted value range: -1 <= xOffset <= 1
IS_GET_KEYOFFSET_X	
yOffset	Y offset, permitted value range: -1 <= yOffset <= 1
IS_GET_KEYOFFSET_Y	

Return value:

In combination with IS_GET_KEYOFFSET_X and IS_GET_KEYOFFSET_Y the current settings are read, otherwise IS_SUCCESS or IS_NO_SUCCESS.

4.46 SetScaler

Syntax:

LONG SetScaler (FLOAT ScaleX, FLOAT ScaleY)

Description:

SetScaler() activates or deactivates the scaler and at the same time sets the horizontal and vertical scaling factor. If both *ScaleX* and *ScaleY* are set to 100 (100%), the scaler is deactivated. The scaler permits scaling factors of up to 1/16 (6.25%) on the horizontal and vertical axis. An integrated interpolation filter suppresses undesirable block artifacts in the image. We recommend deactivating the scaler for image processing.

Transfer parameter:

ScaleX	Horizontal scaling factor in the value range from 6.25 to 100
IS_SET_SCALER_ON	Activate scaler
IS_SET_SCALER_OFF	Deactivate scaler
IS_GET_SCALER_MODE	
ScaleY	Vertical scaling factor in the value range of 6.25 to 100

Return value:

Current setting in combination with IS_GET_SCALER_MODE, otherwise IS_SUCCESS or IS_NO_SUCCESS

4.47 SetSync (EAGLE only)

Syntax:

LONG SetSync (LONG Sync)

Description:

SetSync() programs the sync generator on EAGLE. Sync generator timing is dependent on the video mode (NTSC, PAL, SECAM) previously set with *SetVideoMode()*.

Transfer parameter:

nSync
IS_SET_SYNC_GEN_OFF
IS_SET_SYNC_GEN_ON
IS_GET_SYNC_GEN

Return value:

IS_SUCCESS, IS_NO_SUCCESS or current setting in combination with IS_GET_SYNC_GEN

4.48 SetSyncLevel

Syntax:

LONG SetSyncLevel (LONG Level)

Description:

SetSyncLevel() sets the voltage level at which the synchronisation signal can still be detected. A voltage of 75 mV or 125 mV can be set. The 75 mV setting is particularly useful for critical video signals such as may occur with video recorders or laser discs.

Transfer parameter:

Level

IS_SET_SYNC_75	Set sync voltage to 75 mV
IS_SET_SYNC_125	Set sync voltage to 125 mV
IS_GET_SYNC_LEVEL	Read out the current setting

Return value:

Current setting in combination with IS_GET_SYNC_LEVEL, otherwise IS_SUCCESS or IS_NO_SUCCESS

4.49 SetVertFilter

Syntax:

LONG SetVertFilter (LONG Mode)

Description:

SetVertFilter() activates the vertical interpolation filters. This function is only useful if image scaling is activated. Select a filter level by linking it with IS_ENABLE_VERT_FILTER by means of an OR operator. Only level 1 is permitted for the vertical filter at all resolutions. Level 2 is works up to a horizontal resolution of 385 pixels and level 3 up to 193 horizontal pixels. Example: SetVertFilter (hf, IS_ENABLE_VERT_FILTER | IS_VERT_FILTER_STEP1).

Transfer parameter:

Mode

IS_DISABLE_VERT_FILTER	Deactivate the filter
IS_ENABLE_VERT_FILTER	Activate the filter
IS_VERT_FILTER_STEP1	Filter level 1 (low filtering)
IS_VERT_FILTER_STEP2	Filter level 2
IS_VERT_FILTER_STEP3	Filter level 3 (strong filtering)
IS_GET_VERT_FILTER_MODE	Read out current setting
IS_GET_VERT_FILTER_STEP	Read out current setting

Return value:

Current setting in combination with IS_GET_VERT_FILTER_MODE or IS_GET_VERT_FILTER_STEP, otherwise IS_SUCCESS, IS_NO_SUCCESS

4.50 SetVideoInput

Syntax:

LONG SetVideoInput (LONG Source)

Description:

SetVideoInput() selects the video input to which the video source to be digitised is connected. If the video sources are not synchronised with each other, the time the multiplexer needs to switch over and thus the time until it synchronises with the new video source may amount to several frames. We therefore recommend operating the cameras synchronously. The EAGLE frame grabber has an integrated sync generator for this purpose.
Please note: While the FALCON frame grabber has only three inputs, four cameras can be connected to EAGLE. Therefore it is necessary to distinguish between the parameters for selecting the inputs for FALCON and EAGLE.

FALCON parameters:

Source

IS_SET_VIDEO_IN_1	Video input 1 (Composite Video)
IS_SET_VIDEO_IN_2	Video input 2 (Composite Video)
IS_SET_VIDEO_IN_S	Video input 3 (S-Video; SVHS)
IS_GET_VIDEO_IN	Read out current setting

EAGLE parameters:

Source

IS_SET_VIDEO_IN_1	Video input 1 (Composite Video)
IS_SET_VIDEO_IN_2	Video input 2 (Composite Video)
IS_SET_VIDEO_IN_3	Video input 3 (Composite Video)
IS_SET_VIDEO_IN_4	Video input 4 (Composite Video)
IS_SET_VIDEO_IN_1S	Video input 1 (S-Video; SVHS)
IS_SET_VIDEO_IN_2S	Video input 2 (S-Video; SVHS)
IS_SET_VIDEO_IN_3S	Video input 3 (S-Video; SVHS)
IS_SET_VIDEO_IN_4S	Video input 4 (S-Video; SVHS)
IS_GET_VIDEO_IN	Read out current setting

Return value:

Current setting in combination with IS_GET_VIDEO_IN, otherwise IS_SUCCESS, IS_NO_SUCCESS

4.51 SetVideoMode

Syntax:

LONG SetVideoMode (LONG Mode)

Description:

SetVideoMode() selects the video standard of the connected video source. Colour and monochrome standard signals at 50 Hz and 60 Hz are supported. The video standard of the connected video source can be detected automatically with the parameter IS_SET_VM_AUTO. All required settings, such as for example the digitisation frequency, are then made by the driver. Automatic detection takes approximately one second.

Transfer parameter:

Mode

IS_SET_VM_PAL	PAL standard colour 50 Hz
IS_SET_VM_NTSC	NTSC standard colour 60 Hz
IS_SET_VM_SECAM	SECAM standard 50 Hz
IS_SET_VM_AUTO	Automatic detection of video standard
IS_GET_VIDEO_MODE	Read out current setting

Return value:

Current setting in combination with IS_GET_VIDEO_MODE, otherwise IS_SUCCESS, IS_NO_SUCCESS

4.52 SetVideoSize

Syntax:

LONG SetVideoSize (LONG xpos, LONG ypos, LONG xsize, LONG ysize)

Description:

SetVideoSize() defines the area of the video signal that is to be digitised. This enables adjustment to various sensor sizes. The left and top edge of the image, image width and height are set.

Transfer parameter:

xpos	x-offset of left edge (default: PAL: 156, NTSC: 120) Limit values for xpos+xsize xpos must be an even value
ypos	y-offset of top edge (default: PAL: 33, NTSC: 21 Limit values for ypos+ysize ypos must be an odd value
xsize	image width
ysize	image height

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.53 ShowColorBars

Syntax:

LONG ShowColorBars (LONG Mode)

Description:

ShowColorBars() displays colour bars. Depending on the set colour mode, these bars are either coloured or monochrome. On FALCON these colour bars are generated by a colour bar generator. This function enables you to quickly determine whether the frame grabber runs in your computing environment, as it tests the file path from frame grabber to VGA card. The colour bar generator is basically an on-board video source.

Transfer parameter:

Mode

IS_SET_CBARS_OFF	Deactivate the colour bar generator
IS_SET_CBARS_ON	Activate the colour bar generator
IS_GET_CBARS_MODE	Read out the current setting

Return value:

Current setting in combination with IS_GET_CBARS_MODE, otherwise IS_SUCCESS, IS_NO_SUCCESS

4.54 ShowDDOverlay

Syntax:

LONG ShowDDOverlay ()

Description:

ShowDDOverlay() shows the overlay in DirectDraw back buffer mode. The data last held in the overlay buffer are displayed. The display is now generated through three image buffers. Depending on your VGA card, the image refresh rate may now be lower than without overlay display.

Transfer parameter:

<none>

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.55 StopLiveVideo

Syntax:

LONG StopLiveVideo (LONG Wait)

Description:

The *StopLiveVideo()* function stops live mode and freezes the image in the memory of the VGA card or the PC system memory. The return behaviour of the function is controlled via the *Wait* parameter. You can choose between the two modes "Function returns immediately to the calling function" and "Function waits until the last image has been completely acquired". In the former case, digitisation of the image is completed in the background.

Transfer parameter:

Wait

IS_WAIT	Function waits until the image is entirely in memory
IS_DONT_WAIT	Function returns immediately
IS_FORCE_VIDEO_STOP	Stop digitisation immediately

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.56 UpdateDisplay

Syntax:

LONG UpdateDisplay ()

Description:

UpdateDisplay() manually updates the display in the DirectDraw modes. This update is normally performed automatically by the driver. In a few select cases it may be necessary to manually adjust the display to changed circumstances. (See also function *SetDisplayMode()).*

Transfer parameter:

<none>

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.57 WriteEeprom

Syntax:

LONG WriteEeprom (BSTR string)

Description:

A rewriteable EEPROM is located on the frame grabber as a small storage unit. In addition to the information stored permanently on the card, 64 bytes of own data can be written to the EEPROM. The contents of this 64-byte block can be read out with the *ReadEEPROM()* function. Please note: No more than 64 characters from the passed string will be written to the EEPROM.

Transfer parameter:

string	String containing proprietary data

Return value:

IS_SUCCESS, IS_NO_SUCCESS

5 Initialisation properties

Variables that can only be changed before initialising the ActiveX Control and can also be changed via the Properties dialogue of the Active X Control:

5.1 AllowPopupMenu

Syntax:

BOOL AllowPopupMenu

Description:

AllowPopupMenu allows the user to call up the popup menu with the right mouse button. It can then be used to make grabber settings at runtime. In addition, the Properties dialogue can be opened.

5.2 AllowDirectDraw

Syntax:

BOOL AllowDirectDraw

Description:

The ActiveX Control uses two options to show the live image, which are also described in the manual for the grabber: BitmapMode or DirectDraw. While pure bitmap display works on any PC, but is slow, the PC and particularly the graphics card must meet certain requirements for DirectDraw mode. Please note the information in the manual for the frame grabber.
If *AllowDirectDraw* is set to TRUE, the ActiveX Control will attempt to initialise DirectDraw mode when it is initialised (back buffer mode with permission to swap the DirectDraw buffer to the system memory). If DirectDraw is not supported or cannot be initialised, bitmap display is automatically activated.

5.3 InitBrightness

Syntax:

LONG InitBrightness

Description:

InitBrightness sets the brightness value on initialisation (see function SetBrightness()). -1 means: Default setting is used.

5.4 InitContrast

Syntax:

LONG InitContrast

Description:

InitContrast sets the contrast value on initialisation (see function SetContrast()). -1 means: Default setting is used.

5.5 EnableErrorReport

Syntax:

BOOL EnableErrorReport

Description:

EnableErrorReport activates (1) or deactivates (2) automatic display of error messages. See function SetErrorReport().

5.6 InitHue

Syntax:

LONG InitHue

Description:

InitHue sets the hue value on initialisation (see function SetHue()). -1 means: Default setting is used.

5.7 InitImageHeight

Syntax:

LONG InitImageHeight

Description:

InitImageHeight sets the image height on initialisation. -1 means: Default setting is used (480 for NTSC, 576 for PAL)

5.8 InitImageWidth

Syntax:

LONG InitImageWidth

Description:

InitImageWidth sets the image width on initialisation. -1 means: Default setting is used (640 for NTSC, 768 for PAL)

5.9 InitAtStartup

Syntax:

BOOL InitAtStartup

Description:

InitAtStartup determines whether the ActiveX Control is initialised immediately after the window is created (TRUE, 1) or not (FALSE, 0). For correct operation it must be ensured that the window in which the ActiveX Control is embedded already exists when the ActiveX Control is initialised. Problems in this respect may particularly occur with Visual Basic. Please note the examples supplied and the right time for initialising.

5.10 LiveAtStartup

Syntax:

BOOL LiveAtStartup

Description:

LiveAtStartup determines whether a live image is to be displayed immediately on startup (TRUE, 1) or not (FALSE, 0).

5.11 InitNTSC

Syntax:

BOOL InitNTSC

Description:

InitNTSC determines whether the grabber is to be switched to NTSC mode immediately on initialisation. See also the function SetVideoMode().

5.12 InitSaturationU

Syntax:

LONG InitSaturationU

Description:

InitSaturationU sets the colour value of the U signal on initialisation (see function SetSaturation()). -1 means: Default setting is used.

5.13 InitSaturationV

Syntax:

LONG InitSaturationV

Description:

InitSaturationV sets the colour value of the V signal on initialisation (see function SetSaturation()). -1 means: Default setting is used.

5.14 InitScalerOn

Syntax:

BOOL InitScalerOn

Description:

InitScalerOn sets whether the scaler is activated immediately after the grabber is initialised (TRUE, 1) or not (FALSE, 0). See also function SetScaler() with the parameters ScaleX = 1 and ScaleY = 0.

5.15 Show

Syntax:

BOOL Show

Description:

Show determines whether an image is to be displayed (TRUE, 1) or not (FALSE, 0) in bitmap mode (see AllowDirectDraw and the SetDisplayMode() function).

5.16 InitVideoInput

Syntax:

LONG InitVideoInput

Description:

InitVideoInput determines the input channel on initialisation (see function SetVideoInput()).

