SNAPPER ActiveX Library

Programmer's Manual

DataCell Limited

v4.0.1

Disclaimer

Copyright Notice

Trademarks

Part Information

Part Number: SNP-MAN-ACTIVEX-LIB

Version v4.0.1 September 1999

Printed in the United Kingdom.

Contact Details

Europe & ROW	Web	www.datacell.co.uk
	Sales	info@datacell.co.uk
	Support	techsupport@datacell.co.uk
USA	Web	www.datacell.com
	Sales	info@datacell.com
	Support	techsupport@datacell.com

Head Office:
DataCell Limited.
Falcon Business Park, 40 Ivanhoe Road,
Finchampstead, Berkshire, RG40 4QQ, UK

Tel +44 (0) 1189 324324
Fax +44 (0) 1189 324325

Table of Contents

Overview

SnapperTool is just one example of an *ActiveX client* application created using the ActiveX SDK. The power behind SnapperTool actually lies within a suite of *ActiveX Automation Servers*. SnapperTool makes calls to these servers in order to display a live image into a window, freeze an image, display the configuration sheets and to save and load the configuration files.

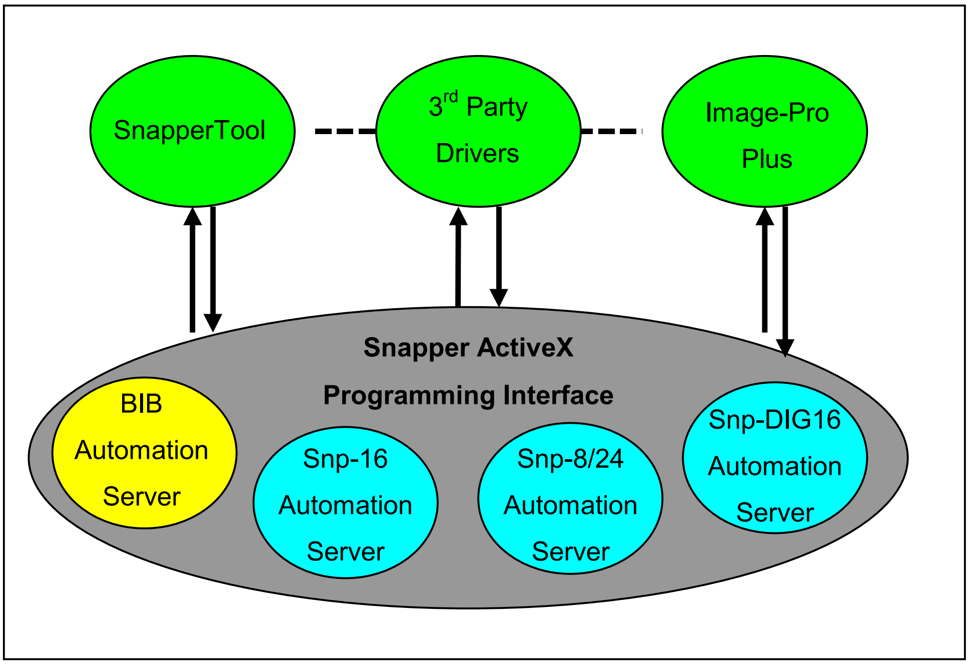

SnapperTool
3rd Party Drivers
Image-Pro Plus
Snapper ActiveX Programming Interface
BIB Automation Server
Snp-16 Automation Server
Snp-8/24 Automation Server
Snp-DIG16 Automation Server

Figure 1 Snapper ActiveX Programming

The **BIB** (or Bus Interface Board) ActiveX server is only used to initialise the board and identify which of the Snapper ActiveX servers needs to be started.

Each of the different types of acquisition board Snapper-DIG16, Snapper-16 or Snapper24/8 has its own **ActiveX Automation Server**, but the programming interface to each is consistent, i.e. if you want to capture a single image you make a "Snap" call to the server regardless of whether it is a Snapper-DIG16 or Snapper-24. The flexibility and functionality of these servers can be integrated as part of any third-party application using only a few procedure calls.

These consist of

Grab	Displays a live image into the preview window
Snap	Captures an image to memory
Freeze	Halts the live display
LoadConfigFile	Loads in a SnapperTool configuration file
SaveConfigFile	Saves a SnapperTool configuration file
SetParameter	Updates the Snapper configuration
GetParameter	Reads back the Snapper configuration
ShowPropertySheet	Displays configuration dialog
ShowTechnicalSupportSheet	Shows Help information

The *SetParameter* is used so that the application can have its own customised Graphic User Interface (GUI) and still use the Snapper ActiveX servers.

The *GetParameter* is only used to read back the configuration in order to set the customised GUI.

Alternatively the application can use the existing configuration dialogs by calling *ShowPropertySheet*.

Each of these calls is explained in detail in the remainder of this manual. The next section shows example code for initialising these automation servers and employing them to grab images. This code is taken from the example Visual C++ program provided with the ActiveX SDK.

Sample C/C++ Applications

Since there is an automation server for every type of Snapper the first step in using the ActiveX SDK is to start the BIB automation server. This will attempt to locate a Snapper baseboard and initialise the Snapper. This function will return with a handle to the baseboard as well as a reference to the type of snapper (via i32ModuleID).

```
   if (!m_BIBInterface.CreateDispatch("BIB.AUTOMATIONINTERFACE")){
   AfxMessageBox("Couldn't create bib object");
   return;
}

long i32BaseIndex = BASE_AUTO;  //Automatically search for PCI
long i32ModuleID = 0; // Look for any type of SNP module
m_hBase = m_BIBInterface.GetBIBHandle(&i32BaseIndex, &i32ModuleID, TRUE);
```

Having successfully located a Snapper board the next stage is to start the appropriate Snapper automation server. The code below simply Creates the appropriate automation server given the information returned in i32ModuleID.

```
if (m_hBase != NULL) {
   switch (i32ModuleID) {
      case SNP24_ID:
      case SNP8_ID : //Snapper24/8 defined in asl_inc.h
         if (!m_SnapperInterface.CreateDispatch("SNAP8.CoreInterface")){
            AfxMessageBox("Couldn't create Snapper object");
            return;
         }
         break;

      case SNP16_ID : //Snapper16 defined in asl_inc.h
         if (!m_SnapperInterface.CreateDispatch("SNAP16.CoreInterface")){
            AfxMessageBox("Couldn't create Snapper object");
            return;
         }
         break;

      case DIG16_ID :  //Snapper DIG16 defined in asl_inc.h
         if (!m_SnapperInterface.CreateDispatch("SNAPDIG16.CoreInterface")){
            AfxMessageBox("Couldn't create Snapper object");
            return;
         }
         break;
   }
}
```

The Snapper automation server has been started and is now ready for use by the application.

CAPTURING IMAGES

Having initialised the Snapper ActiveX servers the most common requirement is to display live within an image. The following ***Grab*** function call will start live-acquisition and display to a particular window.

```
struct StatusBarValues  st;
Thandle Image;
…..
m_snapperInterface.Grab( m_hBase,  m_hWnd, &st);
….
Image = m_snapperInterface.Snap( m_hBase);
…….
```

This image can be frozen at any time using the ***Snap*** call which returns an image handle. This handle can be used to store the image to disk or processed using the functions provided in the TMG library (see Snapper SDK manual for details).

For example the code below snaps an image using the Automation servers and uses functions from the main Snapper SDK to save the image.

```
Image = m_snapperInterface.Snap( m_hBase);
TMG_image_set_outfilename(Image, “.\\test.bmp”);
TMG_image_write(Image, TMG_NULL, TMG_BMP,TMG_RUN);
…….
```

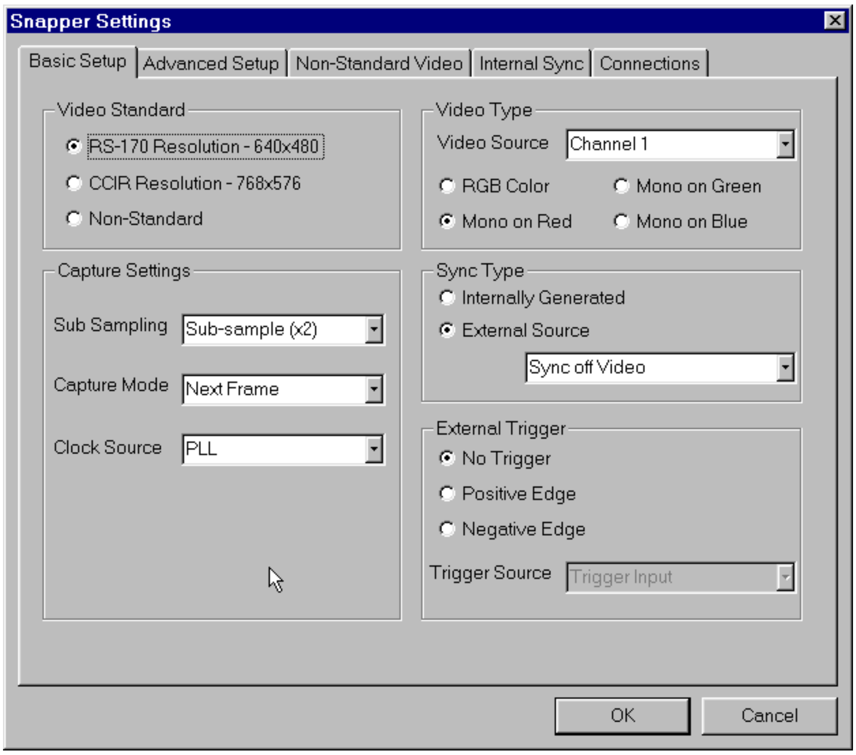
For details of other operations you may perform on Snapper image handles consult the TMG section of the Snapper programming manual.

DISPLAYING DIALOG BOXES

Any of the Configuration, Display modes, or Help dialog boxes can be invoked with a single call e.g.

```
m_snapperInterface.ShowPropertiesDialog(m_hBase,m_hWnd);
```

This will present the user with the configuration dialog box for that type of Snapper. For example the Snapper24 properties dialog box is shown below



Video Standard
● RS-170 Resolution - 640x480
○ CCIR Resolution - 768x576
○ Non-Standard

Capture Settings
Sub Sampling: Sub-sample (x2)
Capture Mode: Next Frame
Clock Source: PLL

Video Type
Video Source: Channel 1
○ RGB Color
○ Mono on Green
● Mono on Red
○ Mono on Blue

Sync Type
○ Internally Generated
● External Source
Sync off Video

External Trigger
● No Trigger
○ Positive Edge
○ Negative Edge
Trigger Source: Trigger Input

This can be used to change the configuration of the Snapper even while the live video is being updated. For more details of the SnapperTool dialogs consult the SnapperTool manual or on-line help.

CHANGING PARAMETER SETTINGS

A complete application with the ability to configure Snapper for any type of video device can be completed using the ActiveX SDK within minutes. However if the application requires a completely customised interface then this can also be provided using the *SetParameter*and *GetParameter*calls

```
width = m_snapperInterface.GetParameter(hBase,"Active Area X Length");
m_snapperInterface.SetParameter(hBase,"Active Area X Length",width+1);
```

The Customised user interface can make live updates to the Snapper configuration using these calls. The Image-Pro Plus driver and the LabVIEW drivers are examples of customised GUIs using the ActiveX SDK frame work.

Sample Visual Basic Applications

Creating Visual Basic programs is even easier than building Visual C++ applications since the ability to use ActiveX objects is inherent to Visual Basic. The example program provided with the SDK shows both how to initialise the board and make function calls. An extract of the code is shown below

```
Dim bibobject As Object  ' ActiveX BIB object
Dim snapobject As Object ' ActiveX Snapper object
Dim hBIB As Long         ' Base board address
…..
Private Sub form_Load()
Dim BaseAddress As Long
Dim SnapperID As Long
Dim ShowDialog As Boolean

   BaseAddress = 20    ' BASE_AUTO from asl_inc.h
   SnapperID = 0       ' Search for any type of Snapper
   ShowDialog = 1
   Set bibobject = CreateObject("BIB.AUTOMATIONINTERFACE")
   Set hBIB = bibobject.GetBIBHandle(BaseAddress, SnapperID, ShowDialog)
      If SnapperID = 176 Then         ' Snapper 24
         Set snapobject = CreateObject("Snap8.coreInterface")
      ElseIf SnapperID = 184 Then     ' Snapper 8
         Set snapobject = CreateObject("Snap8.coreInterface")
      ElseIf SnapperID = 160 Then     ' Snapper 16
         Set snapobject = CreateObject("Snap16.coreInterface")
      ElseIf SnapperID = 192 Then     ' Snapper DIG16
         Set snapobject = CreateObject("SnapDIG16.coreInterface")
      End If
   ReturnValue = snapobject.LoadConfigFile(hBIB, ".\VB Example.snp")
End Sub
```

This is equivalent to the C++ code which switches between the different types of Snapper to initialise the Snapper.

```
Private Sub Command2_Click()
   ReturnValue = snapobject.Grab(hBIB, Form1.hWnd, 0)
End Sub
```

A simple button click can start the live display of an image to the forms window with just one function call. For more details look at the Visual basic example form supplied.

Function List

INITIALISATION FUNCTIONS

GetBIBHandle

CONFIGURATION FUNCTIONS

SetParameter
GetParameter
GetServerInfo
LoadConfigFile
SaveConfigFile

DIALOG BOXES

ShowPropertySheet
ShowDisplayModesDialog
ShowTechnicalSupportSheet

IMAGE CAPTURE FUNCTIONS

Snap
Grab
Freeze

The functions are described in alphabetical order in the following pages.

GetBIBHandle

USAGE

Thandle GetBIBHandle (long FAR pBaseIndex, long FAR *pSnapperID, BOOL bDisplayDialog)*

ARGUMENTS

pBaseIndex	Pointer to the SnapperTool Base address index number to look for board.
	0 First available PCI board 1 1st PCI board 2 2nd PCI board 3 3rd PCI board 4 4th PCI board 5 ISA board 0x300 6 ISA board 0x320 7 ISA board 0x340 8 ISA board 0x380 9 ISA board 0x3A0 10 ISA board 0x3C0 11 ISA board 0x3E0
	The value in the pointer is updated upon return.
pSnapperID	long ptr to the ID of snapper to look for; One of the defined constants:
	SNP8_ID Snapper-24 or Snapper-8 *SNP24_ID* Snapper-24 *SNP16_ID* Snapper-16 *DIG16_ID* Snapper-DIG16 *NULL* Any Snapper value is returned
bDisplayDialog	Boolean flag to indicate whether the Dialog should be displayed in the event of any errors

DESCRIPTION

The method attempts to look for a board at the address passed in *pBaseAddress*. If successful it returns a handle to the board.

If there is no available board at this address, a base address selection dialogue is displayed. The method will cycle until a board is found or the dialogue is closed or cancelled. The dialogue box can be suppressed by setting *bDisplay* to FALSE.

RETURNS

This returns a Handle (*Thandle*) to the Board found. If no board is found then the function returns *NULL*.

SEE ALSO

ShowPropertySheet, *ShowDisplayModesDialog*, *ShowTechnicalSupportSheet*, *LoadConfigFile*, *SaveConfigFile*, *Serialize*, *Grab*, *Snap*, *Freeze*

ShowPropertySheet

USAGE

long ShowPropertySheet (long hBIB, long hWindow)

ARGUMENTS

hBIB	BIB handle
hWindow	Parent Window handle (HWND), cast to long

DESCRIPTION

The method displays, as a modeless dialog, the Snapper's property sheet.

RETURNS

TRUE if no problem, *FALSE* otherwise.

SEE ALSO

ShowDisplayModesDialog, *GetBIBHandle*

ShowDisplayModesDialog

USAGE

long ShowDisplayModesDialog (long hBIB, long hWindow)

ARGUMENTS

hBIB	BIB handle.
hWindow	Parent Window handle (HWND), cast to long.

DESCRIPTION

The method displays, as a modeless dialog, the Acquisition Modes dialog. Some modes may not be available dependent on the nature of the graphics card drivers installed and the baseboard selected (not all modes are supported with ISA-BIBs).

RETURNS

The display mode used:

0	DIB
1	DDB
2	DirectDraw
3	DirectDraw using Sequence mode

SEE ALSO

GetBIBHandle

ShowTechnicalSupportSheet

USAGE

long ShowTechnicalSupportSheet (long hBIB, long hWindow)

ARGUMENTS

hBIB	BIB handle.
hWindow	Parent Window handle (HWND), cast to long

DESCRIPTION

The method displays, as a modeless dialog, the Snapper's environment sheet.

RETURNS

TRUE if there is no problem and *FALSE* otherwise.

SEE ALSO

GetBIBHandle

Grab

USAGE

long Grab (long hBIB, long hWindow, long pStatus)

ARGUMENTS

hBIB	BIB handle.
hWindow	Parent Window handle (HWND), cast to long
pStatus	*NULL* or Pointer to a Status structure, cast to long.

DESCRIPTION

The method attempts to display continuous live video. If there is a problem with the requested display mode, DIB will be used. The Status structure will be updated regularly until Grab returns.

The Status structure is defined as:

```
struct StatusBarValues
{
   long  m_i32SizeX;      //Width of image captured
   long  m_i32SizeY;      //Height of image captured
   long  m_i32SubSample;  //Subsample of image captured
   float m_f32FPS;        //Frames displayed per second
};
```

For no status information set *pStatus* to *NULL*.

RETURNS

The display mode used by SnapperTool to display the live image.

0	DIB
1	DDB
2	DirectDraw
3	DirectDraw using Sequence mode

SEE ALSO

Freeze, *Snap*

Snap

USAGE

long Snap (long hBIB)

ARGUMENTS

hBIB	BIB handle.

DESCRIPTION

The method captures and returns a single image.

RETURNS

TMG Handle (*Thandle*) of the snapped image.

SEE ALSO

Grab, *Freeze*

Freeze

USAGE

Thandle Freeze(long hBIB)

ARGUMENTS

hBIB	BIB handle.

DESCRIPTION

The method freezes the live image and returns the frozen image.

RETURNS

This functions returns a handle (*Thandle*) to an image structure containing the frozen image. The image format returned will be either RGB, Y8, or Y16.

SEE ALSO

Grab, *Snap*

GetServerInfo

USAGE

long GetServerInfo (LPCTSTR InfoString)

ARGUMENTS

InfoString	String describing the version information to be returned

DESCRIPTION

The method returns version info about the requested component. Current options from the server include

"ServerVersion"	The numerical version of the Server.
"SDKVersion"	The version of the SDK the server was built with.
"BIBRevision"	The version of the BIB the server was built with.
"SnapperRevision"	The version of the Snapper the server was built with.
"OSVersion"	The ID of the Operating System the server was built for: 1 Windows 95 2 Windows NT (Not available at present) 3 Mac System7 (Not available at present)

RETURNS

Requested version numbers.

SEE ALSO

Grab, *Snap*

LoadConfigFile

USAGE

long LoadConfigFile (long hBIB, LPCTSTR FileName)

ARGUMENTS

hBIB	BIB handle.
FileName	File name of a SnapperTool configuration file.

DESCRIPTION

The method reads the configuration file and sets up the Snapper accordingly. If the file cannot be opened, default settings will be used.

RETURNS

TRUE if there is no problem and *FALSE* otherwise.

SEE ALSO

SaveConfigFile

SaveConfigFile

USAGE

long SaveConfigFile (long hBIB, LPCTSTR FileName)

ARGUMENTS

hBIB	BIB handle.
FileName	File name of a SnapperTool configuration file.

DESCRIPTION

The method saves the current setup to the configuration file.

RETURNS

TRUE if there is no problem and *FALSE* otherwise.

SEE ALSO

LoadConfigFile

Serialize

USAGE

long Serialize (long hBIB, long pCArchive)

ARGUMENTS

hBIB	BIB handle.
pCArchive	Pointer to a CArchive object, cast to long

DESCRIPTION

The method saves the current setup to a *CArchive* object. For use with MFC Serialize functions.

RETURNS

TRUE if there is no problem and *FALSE* otherwise.

SEE ALSO

LoadConfigFile

GetParameter

USAGE

long GetParameter (long hBIB, LPCTSTR ParameterName)

ARGUMENTS

hBIB	BIB handle.
ParameterName	String containing parameter to return. (see Description below)

DESCRIPTION

The method returns the string describing the setting of a parameter. For instance in order to find the width of the image Snapper will create you would call

```
…
width = GetParameter(hBib,"Active Area X Length");
…
```

The full parameter options are shown below together with the Snapper modules they work with.

COMMON DISPLAY

These display parameters apply to all Snappers:

Parameter	Value	Description
"Display Mode"	0	DIB
	1	DDB
	2	DDraw
	3	DDraw and sequence mode

LUTs

These LUT related parameters apply to Snapper-DIG16, Snapper-24 and Snapper-8:

Parameter	Value	Description
"Brightness Level"	0-100	Change brightness (0 minimum, 100 maximum)
"Contrast Level"	0-100	Change contrast (0 minimum, 100 maximum)
"Gamma Level"	0-100	Change gamma (0 minimum, 100 maximum)
"Red Intensity"	0-100	Change Red intensity (0 minimum, 100 maximum), Snapper-24 only
"Green Intensity"	0-100	Change Green intensity (0 minimum, 100 maximum), Snapper-24 only
"Blue Intensity"	0-100	Change Blue intensity (0 minimum, 100 maximum), Snapper-24 only

ROI

These Region of Interest (ROI) related parameters apply to Snapper-DIG16, Snapper-24 and Snapper-8:

Parameter	Value	Description
"Active Area X Start"	1-n	Horizontal start of acquisition
"Active Area X Length"	1-n	Horizontal length of acquisition
"Active Area Y Start"	1-n	Vertical start of acquisition
"Active Area Y Length"	1-n	Vertical length of acquisition

DIG16

These parameters only apply to Snapper-DIG16:

Parameter	Value	Description
"Capture Mode"	0	First field
	1	Next field
"Trigger Source"	0	IO_A
	1	IO_B
	2	IO_C
	3	IO_D
	4	TTL_1
	5	TTL_2
"Trigger Mode"	0	No trigger
	1	Positive edge trigger
	2	negative edge trigger
"IO Line A Mode"	0	Input
	1	Output high
	2	Output low
"IO Line B Mode"		As "IO Line A Mode"
"IO Line C Mode"		As "IO Line A Mode"
"IO Line D Mode"		As "IO Line A Mode"
"Output A Mode"	0	Disabled
	1	Output high
	2	Output low
	3	Exposure output enabled
"Output B Mode"	0	Disabled
	1	Output high
	2	Output low
"Output C Mode"		As "Output B Mode"
"Output D Mode"		As "Output B Mode"
"MSB Position"		Most significant bit, in range 0..15
"LSB Position"		Least significant bit, in range 0..15
"Bit Depth"		Image depth, in range 0..15
"Alignment Mode"	0	LSB aligned
	1	MSB aligned
"Clock Source Mode"	0	Positive edge external clock
	1	Negative edge external clock
	2	Positive edge internal clock
	3	Negative edge external clock

"Clock Frequency Mode"	0	251kHz internal clock
	1	501kHz internal clock
	2	1.0MHz internal clock
	3	2.0MHz internal clock
	4	2.5MHz internal clock
	5	3.13MHz internal clock
	6	4.0MHz internal clock
	7	5.0MHz internal clock
	8	6.26MHz internal clock
	9	8MHz internal clock
	10	10MHz internal clock
	11	12.5MHz internal clock
	12	16MHz internal clock
	13	20MHz internal clock
	14	25MHz internal clock
"Continuous Clock Flag"	0	Camera does not provide a continuous clock
	1	camera does provide a continuous clock
"Two Tap Flag"	0	Camera provides a single output
	1	Camera provides two 8 bit channels representing odd/even pixels
"Line Acceptance Mode"	0	Accept every line in line scan mode
	1	Accept every 2nd line in line scan mode
	2	Accept every 4th line in line scan mode
	3	accept every 8th line in line scan mode
"Waterfall Mode"	0	Disable waterfall display mode
	1	Enable waterfall display mode
"Lines Per Bank"		Number of lines to capture per acquisition in line scan mode. This only applies if Waterfall mode is disabled.
"Pulse To Camera Mode"	0	No line pulse
	1	Positive line pulse to drive the camera in line scan mode
	2	Negative line pulse to drive the camera in line scan mode
"Pulse Width"	1-n	Output line width in line scan mode (ms)
"Pulse From Camera Mode"	0	No line pulse
	1	Positive line pulse from the camera in line scan mode
	2	Negative line pulse from the camera in line scan mode
"Data Acquisition Mode"	0	Always acquire Data Stream data
	1	Acquire Data Stream data when Line is high
	2	Acquire Data Stream data when Frame is high
	3	Acquire Data Stream data when Frame and Line are high
"Start Capture Mode"	0	Start Data Stream acquisition immediately
	1	Start Data Stream acquisition when Line is high
	2	Start Data Stream acquisition when Frame is high
	3	Start Data Stream acquisition when both Frame and Line are high
"Comm Standard"	0	RS-422 signal levels
	1	RS-232 signal levels
"Data Bits Mode"	0	7 data bits in serial comms
	1	8 data bits in serial comms
"Stop Bits Mode"	0	1 stop bit in serial comms
	1	2 stop bits in serial comms
"Parity Mode"	0	None
	1	Odd parity
	2	Even parity
	3	Mark
	4	Space

"Speed Mode"	0	9600 baud
	1	14400 baud
	2	19200 baud
	3	28800 baud
	4	57600 baud
"XON\XOFF"	0,1	Disable\Enable XON\XOFF flow control
"Initial File"	0,1	Disable\Enable camera initialisation via comms
"Pregrab File"	0,1	Disable\Enable pre grab serial comms
"Postgrab File"	0,1	Disable\Enable post grab serial comms
"Presnap File"	0,1	Disable\Enable pre snap serial comms
"Postsnap File"	0,1	Disable\Enable post snap serial comms
"Mplus Control Mode"	0	Continuous
	1	Triggered
	2	Computer controlled
"MPlus Exposure Time"	1-	Exposure time only applies to computer controlled MPlus mode
"MPlus Shutter Mode"	0	Shutter runs
	1	Shutter open
"MPlus Frame Reset Mode"	0	Frame expose
	1	Frame reset
"MPlus-i Shutter Mode"	0	Enable shutter
	1	Hold shutter open
	2	Hold shutter closed
"MPlus-i Strobe Mode"	0	Negative strobe polarity
	1	Positive strobe polarity
"MPlus-i Exposure Time"	0-n	Exposure control setting
"MPlus-i Gain"	0-n	Gain settings
"MPlus-i Black Level"	0-n	Black level setting
"MPlus-i Defect Control"	0	Defect control off
	1	Defect control on
"Camera Standard"	0	Use standard camera list
	1	Use non standard camera option
"Standard Camera"	0-n	List of standard cameras (consult the properties panel for the latest list)
"Non Standard Camera"	0	Non standard area scan camera
	1	Non standard line scan camera
	2	Non standard data stream camera
"SubSample Mode"	0	Full frame
	1	Sub sample by 2
	2	Sub sample by 4
	3	Sub sample by 8
"Interlace Mode"	0	Area scan camera is not interlaced
	1	Area scan camera is interlaced
"Integration Mode"	0	Positive exposure pulse
	1	Negative exposure pulse
	2	Double positive pulse
	3	Double negative pulse
"Integration Time"	1-n	Integration period (ms)

SNP16

Parameter	Value	Description

SNP24 & SNP8

These parameters only apply to Snapper-24 and Snapper-8:

Parameter	Value	Description
"Capture Mode"	0	Next frame
	1	Next two fields
	2	Next first field
	3	Next second field
	4	Next field
"Custom lm1882 Value_00" to "Custom lm1882 Value_18"	0–255	See the Snapper-24 SDK for details
"lm1882 Clock Frequency"		See the Snapper-24 SDK for details
"SubSample Mode"	0	Next frame
	1	Next two fields
	2	Next first field
	3	Next second field
	4	Next field
"Video Standard"	0	CCIR
	1	RS170 / EIA
	2	Non Standard
"Video Source"	0-3	Required video input channel
"Sync Mode"	0	Sync off video
	1	Sync of composite
	2	Sync off positive HSync & VSync
	3	Sync of negative HSync & VSync
	4-6	Sync off Red\Green\Blue channel 1
	7-9	Sync off Red\Green\Blue channel 2
	10-12	Sync off Red\Green\Blue channel 3
	13-15	Sync off Red\Green\Blue channel 4
"Sync Source"	0	Internal sync source
	1	External sync source
"Line Frequency"	10-200kHz	Line frequency
"Clock Mode"	1-30MHz	Pixel clock frequency
"PClk Jumper Mode"	0	RS-422 PClk signal levels
	1	TTL PClk signal levels
"Trigger Jumper Mode"	0	RS-422 trigger signal levels
	1	TTL trigger signal levels
"HSync Phase Mode"	0	O degree sampling phase angle
	1	90 degree sampling phase angel
	2	180 degree sampling phase angle
	3	270 degree sampling phase angle
"Sync Output Mode"	0	Composite sync
	1	Negative CSync on HSync
	2	Positive CSync on HSync
	3	Negative HSync and VSync

	4	Positive HSync and VSync
"Trigger Source"	0	Trigger input
	1	HSync input
	2	VSync input
	3	PClk input
"Interlace Mode"	0	Camera is not interlaced
	1	Camera is interlaced
"Integration Mode"	0	Positive exposure pulse
	1	Negative exposure pulse
	2	Double positive pulse
	3	Double negative pulse
"Integration Time"	1-n	Integration period (ms)
"Colour-Mono Input Mode"	0	Full RGB
	1	Mono on Red
	2	Mono on Green
	3	Mono on Blue
		This parameter is not supported for Snapper-8.

RETURNS

The setting of the requested parameter.

SEE ALSO

SetParameter

SetParameter

USAGE

bool SetParameter (long hBIB, LPCTSTR ParameterName, long ParameterValue)

ARGUMENTS

hBIB	BIB handle.
ParameterName	String containing parameter to return.
ParameterValue	Value to set.

DESCRIPTION

This sets the video settings parameters. See *GetParameter* for a list of valid *ParameterNames*.

RETURNS

TRUE if there is no problems and *FALSE* otherwise.

SEE ALSO

GetParameter