VIRCAM SDK
User's Manual

Contents

1. Introduction

VCamServer is a COM interface Server allowing third part software to manage a Cedip camera through the distributed object. VCamServer is a synchronised and multi-threaded process so several client can connect to the camera in parallel.

2. Before Starting

Please note the following conventions used in this document.

This picto indicates a mostly used function

This picto indicates a low level function. **You must not use a function marked with this picto unless you are perfectly aware of the consequences. Some of them can damage the camera.**

TM This symbol designates all words considered to be Trademarks

3. Minimum configuration

The following configuration is required in order to develop you own software:

- PC with Windows 2000 or XP installed
- Developpement tool handling COM interfaces (Visual .NET / Visual C++ 7.1 / Delphi / …)
- Vircam SDK installed
- Vircam software installed

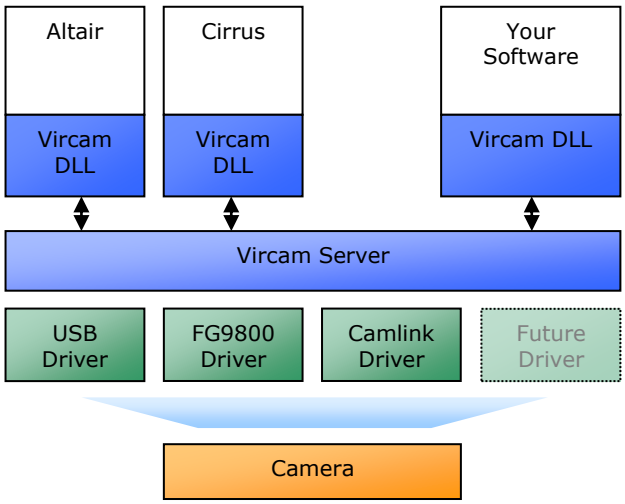
4. What is Vircam SDK

VircamSDK is a developpement library including:

- C++ Libraries (.lib)
- Header files (.h)
- Dynamic Link Librairies (.dll)
- Documentation
- Complete working sample demonstrating mostly used functions.

5. System architecture

Vircam is a combination of a server and a client DLL. Server is in charge of virtualizing all camera interfaces while Client DLL handles the connection to the server.
VircamServer is an ATL/COM component running as a Windows Service. It interfaces all known Cedip camera interfaces and gives a commom entry point to your software, independent from the camera type and interface. The software you will develop using vircamSDK will handle all Cedip cameras. Vircam handles one camera at a time.
The following chart explains the architecture of Cedip's Software.



6. Getting started with Vircam using C++

This section focus only on C++ language.

In order to connect to a camera through Vircam, it is first needed to instantiate the distributed CVirtualCam object.

- Include the following files:

```
#include "_VCamServer_i.c"
#include "_VCamServer.h"
```

- Create an interface pointer:

```
IVirtualCam* m_pVirtualCam;
```

- Create an instance of the CVirtualCam Object:

```
// Initialize COM library
HRESULT hr = CoInitializeEx(NULL, COINIT_MULTITHREADED);
if (FAILED(hr)) //if initialization fails then exit
{
     AfxMessageBox("Failed to initialize COM library");
}

m pVirtualCam = NULL;

//create the com
hr = CoCreateInstance(     CLSID_CVirtualCam, NULL,
                           CLSCTX_LOCAL_SERVER ,IID_IVirtualCam,
                           (void**) &m_pVirtualCam);
```

- Freeing the instance is done as following:

```
if (m_pVirtualCam){
     m_pVirtualCam->Release();
     m_pVirtualCam = NULL;
}
CoUninitialize();
```

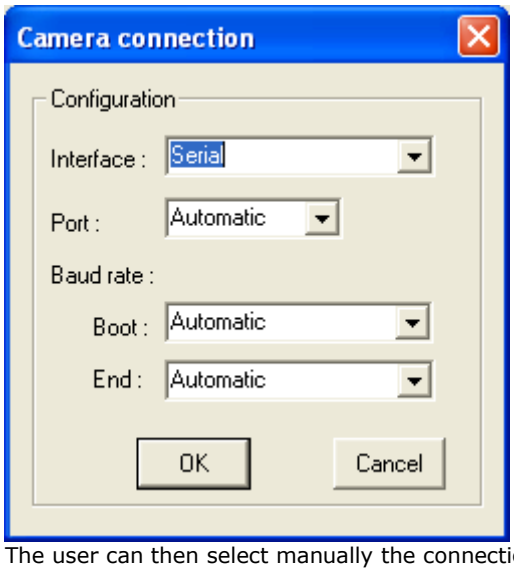
For more details see VirtualClientDll sources.

7. Initialization functions

This section describes functions related to Vircam connection.

7.1. Init

HRESULT Init(ULONG bAuto, ULONG* bReturn)

Function	Connect to the camera and retrieve the internal configuration of the camera
bAuto	Automatic Mode (TRUE/FALSE) When TRUE, Vircam scans all video and communication ports available until it finds a Camera. When FALSE or when scanning failed, the following appears :  The user can then select manually the connection parameters.
bReturn	TRUE if connection successful, FALSE otherwise.

7.2. IsIni

HRESULT IsIni(ULONG* pbInisialized, ULONG* bReturn)

Function	Check if Vircam is already running
pbInisialized	TRUE if Vircam is already running, FALSE Otherwise
bReturn	TRUE if no error occured, FALSE otherwise

7.3. ReInit

HRESULT ReInit(ULONG bAuto, ULONG* bReturn)

Function	Restart Vircam. Used when a connection had been lost to repare it.
bAuto	Automatic Mode (TRUE/FALSE). See *Init*
bReturn	TRUE if no error occured, FALSE otherwise

7.4. IsServerRunning

HRESULT IsServerRunning(void)

Function	Returns S_OK if server ris running.

7.5. GetVersion

HRESULT GetVersion(ULONG* pdwVersionMajeur, ULONG* pdwVersionMineur, ULONG* pdwVersionBuild)

Function	Get Vircam version number
pdwVersionMajeur	Major version
pdwVersionMineur	Minor version
pdwVersionBuild	Build version

8. Debug functions

This section describes functions used for debugging

8.1. SetDebug

HRESULT SetDebug(ULONG bDebug, ULONG* bReturn)

Only for Cirrus platform

Function	Show/Hide debug mode on video output
bDebug	Show/Hide (TRUE/FALSE)
bReturn	TRUE if no error occured, FALSE otherwise

8.2. GetDebug

HRESULT GetDebug(ULONG* pbDebug, ULONG* bReturn)

Only for Cirrus platform

Function	Retreive debug mode state
pbDebug	Mode debug actif (TRUE/FALSE)
bReturn	TRUE if no error occured, FALSE otherwise

8.3. SetActiveSpy

HRESULT SetActiveSpy(ULONG bActive)

Function	Show/Hide communication spy window
bActive	Show/Hide (TRUE/FALSE)

8.4. Command

HRESULT Command(BSTR strCmd, BSTR* pstrAnswer, ULONG* bReturn)

Refer to Cirrus Command Protocol documentation for more information

Function	Send Command
strCmd	BSTR (Basic string) containing the command line
pstrAnswer	BSTR* (pointeur Basic string) containing the answer from the camera.
bReturn	TRUE if no error occured, FALSE otherwise

8.5. SetLogFile

HRESULT SetLogFile(BSTR strLogFile, ULONG dwSizeMax)

Refer to Cirrus Command Protocol documentation for more information

Function	Set file path for saving the spy window communication
strLogFile	BSTR (Basic string) containing the destination file path
dwSizeMax	Maximum destination file size

9. Error Management functions

This section describes functions related to error management

9.1. GetLastError

HRESULT GetLastError(ULONG* pdwError, BSTR* pstrReason, ULONG* bReturn)

Function	Retreive last error
pdwError	Last error code
pstrReason	BSTR (Basic string) containing the error text
bReturn	TRUE if no error occured, FALSE otherwise

10. Image acquisition functions

This section describes functions related to image acquisition.

10.1. IsGrabberCap

HRESULT IsGrabberCap(ULONG* pbCandoIt, BSTR* pstrReason, ULONG* bReturn)

Function	Check if frame grabber is available
pbCandoIt	TRUE if available (TRUE/FALSE)
pstrReason	BSTR (Basic string) string containing the error explanation
bReturn	TRUE if no error occured, FALSE otherwise

10.2. GetImage

HRESULT GetImage(ULONG dwPoint, ULONG dwLine, ULONG dwModulo, BSTR strMem, ULONG dwOffsetImage, BSTR strStructInf, ULONG dwOffsetInfo, ULONG dwTimeOut, ULONG* bReturn)

Function	Acquire one image and its embedded information structure
dwPoint	Number of point of the image
dwLine	Number of line of the image
dwModulo	Number of image per acquisition packet
strMem	BSTR (Basic string) name of shared memory containing the image
dwOffsetImage	Image memory offset
strStructInf	BSTR (Basic string) name of shared memory containing image informations
dwOffsetInfo	Image information memory offset
dwTimeOut	Time out in milliseconds
bReturn	TRUE if no error occured, FALSE otherwise

10.3. GetCurrentTimeCounter

HRESULT GetCurrentTimeCounter(ULONGLONG* pTimeCounter, ULONG* bReturn)

Function	Retreive the relative grabbing time associated to the current acquired image.
pTimeCounter	64bits time (in µs)
bReturn	TRUE if no error occured, FALSE otherwise

10.4. IsModuloImp

HRESULT IsModuloImp(ULONG* pbModulo, ULONG* bReturn)

Function	Check that system as Modulo capability
pbModulo	TRUE if is modulo available
bReturn	TRUE if no error occured, FALSE otherwise

10.5. GetNumberImagePacket

HRESULT GetNumberImagePacket(ULONG* pdwNbrImgInPack, ULONG* pbReturn)

Function	Get image packet number
pdwNbrImgInPack	Number of frame in the packet
pbReturn	TRUE if no error occured, FALSE otherwise

10.6. StopAcqui

HRESULT StopAcqui(ULONG* bReturn)

Function	Stop acquisition
bReturn	TRUE if no error occured, FALSE otherwise

10.7. Image information structure description

Data Type	Offset (bytes)	Designation & Size	Description	Example
WORD	**0**	wStructSize	Structure size	
DWORD	**1**	dwStructCaps	Structure capabilities	Image info valid: 0x00000001 Lockin info valid: 0x00000002 Image info general error: 0x80000000 Multi IT info error: 0x40000000
QWORD	**3**	qwImageTime	Image time (µs)	
QWORD	**7**	qwImageCounter	Image number	
WORD	**11**	wMultiItNumber	Multi IT number	
WORD	**12**	wAnalogicalVideoITChannel	Selected video IT channel (1 based index)	
WORD	**13**	wNumericalVideoITChannel	Not used	
WORD	**14**	wNucIndex	Current image NUC index (1 based index)	
float	**15**	fDetectorTemp	Detector temperature (K)	
float	**17**	fSensorTemp [4]	Housing temperature (K)	
float	**25**	fMasterClock	Master clock (Hz)	
DWORD	**27**	dwIntegrationTime	Integration time(µs)	
float	**29**	fFrameRate	Frame rate (Hz)	
BYTE	**31**	byFilterPosition	Filter position (0 based index)	
bool	**32**	bExternalTrigger	External trigger	
bool	**33**	bDataDisable	Digital date disable	
BYTE	**34**	bySpare[15]	Spare	
WORD	**49**	wLockinBoxVersion	Lockin version	
DWORD	**50**	dwLockinPeriod	Lockin period (seconds)	
DWORD	**52**	dwLockinPhase	Lockin phase	
bool	**54**	bLockinMin	Minimum lockin	
bool	**55**	bLockinMax	Maximum lockin	
BYTE	**56**	bySignalNumber	Number of signal	
WORD	**57**	wLockinMeanValue[4]	4 mean values signals	
DWORD	**65**	dwLockinSpare[2]	Lockin spare	

10.8. How to use shared memory

Image and informations are transferred through shared memory blocks. These blocks are named, and this name is used into GetImage function.

10.8.1. Initialisation example

```
//rand name buffer
CString strRandNumer = "";
srand((unsigned)time( NULL ));

for(int i = 0; i< 10000; i++){
	if (OpenFileMapping(FILE_MAP_READ | FILE_MAP_WRITE, FALSE, m_strSharedName) == NULL)
		break;
	strRandNumer.Format("%d", rand());
	m_strSharedName += strRandNumer;
}

m_Handle = CreateFileMapping (INVALID_HANDLE_VALUE, NULL, PAGE_READWRITE, 0,
				sizeof(WORD)*dwSize*IMAGE_MODULO, m_strSharedName);

if(!m_Handle)
	return FALSE;

m_pwBuffer = (WORD*)MapViewOfFile( m_Handle, FILE_MAP_READ | FILE_MAP_WRITE, 0, 0, 0);
if(!m_pwBuffer){
	CloseHandle(m_Handle);
	return FALSE;
}
```

10.8.2. Memory free example

```
//Image
if (m_pwBuffer){
	if(!UnmapViewOfFile(m_pwBuffer))
		return FALSE;
}

if (m_Handle){
	if(!CloseHandle(m_Handle))
	return FALSE;
}
```

11. Frame rate functions

This section describes functions related to Vircam frame rate.

11.1. SetFrequency

SetFrequency(FLOAT fFrequency, ULONG bAutoFrequency, ULONG* bReturn)

Function	Set frame rate
fFrequency	Wanted frequency
bAutoFrequency	TRUE to automatically change frequency if necessary
bReturn	TRUE if no error occured, FALSE otherwise

11.2. GetFrequency

GetFrequency(FLOAT* pfFrequency, ULONG* bReturn)

Function	Get current frame rate
pfFrequency	Current frequency
bReturn	TRUE if no error occured, FALSE otherwise

11.3. GetMaxFrequency

GetMaxFrequency(FLOAT* pfMaxFrequency, ULONG* bReturn)

Function	Get maximum frame rate
pfMaxFrequency	Maximum frequency
bReturn	TRUE if no error occured, FALSE otherwise

11.4. IsItWorksFloatFrequency

IsItWorksFloatFrequency(ULONG* pbOk, ULONG* bReturn)

Function	Float frame rate capability
pbOk	TRUE if float is allowed for frame rate, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

12. Integration time functions

This section describes functions related to Vircam integration time.

12.1. GetIT

GetIT(LONG nChannel, ULONG* pdwIt, ULONG* pdwDelay, ULONG* bReturn)

Function	Get integration time and delay by channel
nChannel	Wanted channel
pdwIt	Current integration time
pdwDelay	Current delay (not used)
bReturn	TRUE if no error occurred, FALSE otherwise

12.2. SetIntegration

SetIntegration(ULONG dwIntegration, ULONG dwDelay, LONG nChannel, ULONG bAutoFrequency, ULONG* bReturn)

Function	Set integration time
dwIntegration	Wanted integration time
dwDelay	Wanted delay (not used)
nChannel	Wanted channel
bAutoFrequency	TRUE to auto change frequency if necessary, FALSE otherwise
bReturn	TRUE if no error occurred, FALSE otherwise

12.3. IsMultiTI

IsMultiTI(ULONG* bReturn)

Function	Multi Integration time available
bReturn	TRUE if multi IT available, FALSE otherwise

12.4. GetCurrentIT

GetCurrentIT(ULONG* pdwIT, ULONG* pdwDelay, ULONG* bReturn)

Function	Get current integration time and delay
pdwIT	Current integration time
pdwDelay	Current delay (not used)
bReturn	TRUE if no error occurred, FALSE otherwise

12.5. SetCurrentIT

SetCurrentIT(ULONG dwIT, ULONG bAutoFrequency, ULONG* bReturn)

Function	Set current integration time
dwIT	Wanted integration time
bAutoFrequency	TRUE to auto change frequency if necessary, FALSE otherwise
bReturn	TRUE if no error occurred, FALSE otherwise

12.6. GetMultiNbrChannel

GetMultiNbrChannel(LONG* pnbrMultiTI, ULONG* bReturn)

Function	Get activated number of channel
pnbrMultiTI	Current number of channel activated
bReturn	TRUE if no error occurred, FALSE otherwise

12.7. GetMaxMultiTIChannel

GetMaxMultiTIChannel(LONG* pnMax, ULONG* bReturn)

Function	Get multi integration time number of channel
pnMax	Current number of channel
bReturn	TRUE if no error occurred, FALSE otherwise

12.8. SetMultiTi

SetMultiTi(LONG nbrMultiTI, ULONG* bReturn)

Function	Set activated number of channel
nbrMultiTI	Wanted number of channel activated
bReturn	TRUE if no error occured, FALSE otherwise

12.9. GetCurrentMultiTiVideo

GetCurrentMultiTiVideo(LONG* pnCurrent, ULONG* bReturn)

Function	Get current channel
pnCurrent	Current video channel
bReturn	TRUE if no error occured, FALSE otherwise

12.10. SetCurrentMultiTiVideo

SetCurrentMultiTiVideo(LONG nCurrent, ULONG* bReturn)

Function	Set video channel
nCurrent	Wanted video channel
bReturn	TRUE if no error occured, FALSE otherwise

12.11. IsSpi

IsSpi(ULONG* bReturn)

Function	Spectro imaging state (Orion)
bReturn	TRUE if spectro imaging enable

12.12. SetOrionMode

SetOrionMode(ULONG bOrionMode, ULONG* bReturn)

Function	Set orion mode
bOrionMode	TRUE to activate orion mode, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

12.13. GetOrionMode

GetOrionMode(ULONG* bOrionMode, ULONG* bReturn)

Function	Get orion mode
bOrionMode	TRUE if orion mode is enable
Parameter 2	TRUE if no error occured, FALSE otherwise

13. Windowing functions

This section describes functions related to camera windowing.

13.1. GetCurrentWindow

HRESULT GetCurrentWindow(LONG* nCurrentIndex, ULONG* wWidth, ULONG* wHeigth, ULONG* wLeft, ULONG* wHigth, ULONG* bReturn)

Function	Get current window parameters
nCurrentIndex	Current windowing index description des index
wWidth	Current width
wHeigth	Current heigth
wLeft	Current left
wHigth	Current top
bReturn	TRUE if no error occured, FALSE otherwise

13.2. SetWindow

HRESULT SetWindow(LONG nWnd, ULONG* wWidth, ULONG* wHeigth, ULONG* wLeft, ULONG* wTop, ULONG bAutoFrequency, ULONG* bReturn)

Function	Set current window parameters
nWnd	Wanted windowing index
wWidth	Wanted width (only for random windowing)
wHeigth	Wanted heigth (only for random windowing)
wLeft	Wanted left (only for random windowing)
wTop	Wanted top (only for random windowing)
bAutoFrequency	TRUE to change automatically the frame rate when necessary, FALSE otherwise. Example: When the Frequency is too important for the future windowing.
bReturn	TRUE if no error occured, FALSE otherwise

13.3. GetWindow

HRESULT GetWindow(LONG nWnd, ULONG* wWidth, ULONG* wHeigth, ULONG* wLeft, ULONG* wHigth, ULONG* bReturn)

Function	Get window parameter
nWnd	Wanted windowing index
wWidth	Width of the choosed windowing index
wHeigth	Heigth of the choosed windowing index
wLeft	Left of the choosed windowing index
wHigth	Top of the choosed windowing index
bReturn	TRUE if no error occured, FALSE otherwise

13.4. GetRectFull

HRESULT GetRectFull(ULONG* wWidth, ULONG* wHeigth, ULONG* wLeft, ULONG* wHigth, ULONG* bReturn)

Function	Get full windowing parameters
wWidth	Width
wHeigth	Heigth
wLeft	Left
wHigth	Top
bReturn	TRUE if no error occured, FALSE otherwise

13.5. GetRectQuarter

HRESULT GetRectQuarter(ULONG* wWidth, ULONG* wHeigth, ULONG* wLeft, ULONG* wHigth, ULONG* bReturn)

Function	Get Quarter Windowing parameters
wWidth	Width
wHeigth	Heigth
wLeft	Left
wHigth	Top
bReturn	TRUE if no error occured, FALSE otherwise

13.6. GetRectHalf

HRESULT GetRectHalf(ULONG* wWidth, ULONG* wHeigth, ULONG* wLeft, ULONG* wHigth, ULONG* bReturn)

Function	Get Half Windowing parameters
wWidth	Width
wHeigth	Heigth
wLeft	Left
wHigth	Top
bReturn	TRUE if no error occured, FALSE otherwise

13.7. GetRectRandom

HRESULT GetRectRandom(ULONG* wWidth, ULONG* wHeigth, ULONG* wLeft, ULONG* wHigth, ULONG* bReturn)

Function	Get Random Windowing parmeters
wWidth	Width
wHeigth	Heigth
wLeft	Left
wHigth	Top
bReturn	TRUE if no error occured, FALSE otherwise

13.8. GetRandomInfoWindow

HRESULT GetRandomInfoWindow(ULONG* bEnable, ULONG* pwXMin, ULONG* pwXMax, ULONG* pwXStep, ULONG* pwYMin, ULONG* pwYMax, ULONG* pwYStep, ULONG* bReturn)

Function	Get random windowing informations
bEnable	TRUE if random windowing is available
pwXMin	Minimum x coordinates
pwXMax	Maximum x coordinates
pwXStep	X minimum step
pwYMin	Minimum y coordinates
pwYMax	Maximum y coordinates
pwYStep	Y minimum step
bReturn	TRUE if no error occured, FALSE otherwise

13.9. IsEnableHalf

HRESULT IsEnableHalf(ULONG* bReturn)

Function	Is half window available
bReturn	TRUE if available, FALSE otherwise

13.10. IsEnableQuarter

HRESULT IsEnableQuarter(ULONG* bReturn)

Function	Is quarter window available
bReturn	TRUE if available, FALSE otherwise

13.11. IsEnableRandom

HRESULT IsEnableRandom(ULONG* bReturn)

Function	Is random window available
bReturn	TRUE if available, FALSE otherwise

13.12. GetCurrentIndex

HRESULT GetCurrentIndex(LONG* pnCurrent, ULONG* bReturn)

Function	Get current windowing index
pnCurrent	Current windowing index
bReturn	TRUE if no error occured, FALSE otherwise

14. Shutter functions

This section describes functions related to camera shutter.

14.1. IsShutterAvailable

HRESULT IsShutterAvailable(ULONG* pbAvailable, ULONG* bReturn)

Function	Is shutter available
pbAvailable	If TRUE shutter is available, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

14.2. SetShutter

HRESULT SetShutter(ULONG bShutter, LONG nFreq, ULONG* bReturn)

Function	Set shutter parameters
bShutter	TRUE to enable shutter
nFreq	Wanted frequency for shutter action
bReturn	TRUE if no error occured, FALSE otherwise

14.3. GetShutter

HRESULT GetShutter(ULONG* pbShutter, LONG* pnFrequency, ULONG* bReturn)

Function	Gets hutter parameters
pbShutter	TRUE if shutter enable, FALSE otherwise
pnFrequency	Current frequency
bReturn	TRUE if no error occured, FALSE otherwise

14.4. CanHaveShutterState

HRESULT CanHaveShutterState (ULONG* pbReturn)

Function	Get Shutter state capability
bReturn	TRUE if it is available, FALSE otherwise

14.5. GetShutterDown

HRESULT GetShutterDown (ULONG* pbReturn)

Function	Get Shutter state
bReturn	TRUE if it is down, FALSE otherwise

14.6. SetShutterDown

HRESULT SetShutterDown (ULONG bShutterDown, ULONG* pbReturn)

Function	Set shutter position
bShutterDown	Wanted state TRUE for shutter down, FALSE for up
bReturn	TRUE if no error occured, FALSE otherwise

15. Filter Wheel functions

This section describes functions related to Vircam filter wheel.

15.1. SetCurrentFilter

SetCurrentFilter(ULONG dwFilter, ULONG* bReturn)

Function	Select a filter
dwFilter	Wanted filter index
bReturn	TRUE if no error occured, FALSE otherwise

15.2. GetCurrentFilter

GetCurrentFilter(ULONG* pdwFilter, ULONG* bReturn)

Function	Get current filter index
pdwFilter	Current filter index
bReturn	TRUE if no error occured, FALSE otherwise

15.3. GetNbrFilter

GetNbrFilter(ULONG* pdwNbrFilter, ULONG* bReturn)

Function	Get number of filter
pdwNbrFilter	Current number of filter
bReturn	TRUE if no error occured, FALSE otherwise

15.4. GetFilterName

GetFilterName(ULONG dwIndex, BSTR strFilterName, ULONG* bReturn)

Function	Get filter name by index
dwIndex	Wanted Filter index
strFilterName	Current filter name
bReturn	TRUE if no error occured, FALSE otherwise

15.5. SetFilterName

GetFilterName(ULONG dwIndex, BSTR strFilterName, ULONG* bReturn)

Function	Set filter name
dwIndex	Wanted Filter index
strFilterName	Wanted filter name
bReturn	TRUE if no error occured, FALSE otherwise

15.6. IsBlackBodyPresent

IsBlackBodyPresent(ULONG* bReturn)

Function	Is black body available
bReturn	TRUE if black body is present , FALSE otherwise

15.7. SetBBPresent

SetBBPresent(ULONG bPresent, ULONG* bReturn)

Function	Set black body available
bPresent	TRUE to set black body present, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

15.8. IsInsideBBPossible

IsInsideBBPossible(ULONG* bReturn)

Function	Possibility for the wheel to contain an internal black body
bReturn	TRUE if the wheel can have a black body, FALSE otherwise

16. Focus functions

This section describes functions related to camera lens focus control.

16.1. CanFocus

CanFocus(ULONG* pbCanFocus, ULONG* bReturn)

Function	Get focus capability
pbCanFocus	TRUE if focus is available, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

16.2. GetFocalType

GetFocalType(ULONG* pdwFocalType, ULONG* bReturn)

Function	Get focal type
pdwFocalType	Focal type : 0 None multi focal 1 Bi focal 2 Quadri focal 3 Tri focal
bReturn	TRUE if no error occured, FALSE otherwise

16.3. SetStep

SetStep(ULONG dwDirection, ULONG dwStep, ULONG* bReturn)

Function	Set position by step first focal
dwDirection	Direction : 1 go backward 2 go forward
dwStep	Pas : 1 small step 2 big step
bReturn	TRUE if no error occured, FALSE otherwise

16.4. SetStep2

SetStep2(ULONG dwDirection, ULONG dwStep, ULONG* bReturn)

Function	Set position by step second focal
dwDirection	Direction : 1 go backward 2 go forward
dwStep	Pas : 1 small step 2 big step
bReturn	TRUE if no error occured, FALSE otherwise

16.5. GetMaxFocusPos

GetMaxFocusPos(ULONG* pdwFocalPos, ULONG* bReturn)

Function	Get maximum focus position
pdwFocalPos	Maximum position
bReturn	TRUE if no error occured, FALSE otherwise

16.6. SetFocal1

SetFocal1(ULONG dwFocal, ULONG dwPos, ULONG* bReturn)

Function	Set focal parameters (first motor)
dwFocal	Wanted focal : 1 no focal 2 50 mm 3 250 mm
dwPos	Wanted position
bReturn	TRUE if no error occured, FALSE otherwise

16.7. SetFocal2

SetFocal2(ULONG dwFocal, ULONG dwPos, ULONG* bReturn)

Function	Set focal parameters (second motor)
dwFocal	Wanted focal : 1 no focal 2 50 mm 3 250 mm
dwPos	Wanted position
bReturn	TRUE if no error occured, FALSE otherwise

16.8. GetFocal1

GetFocal1(ULONG* pdwFocal, ULONG* pdwPos, ULONG* bReturn)

Function	Get first focal parameters
pdwFocal	Current focal : 1 no focal 2 50 mm 3 250 mm
pdwPos	Current position
bReturn	TRUE if no error occured, FALSE otherwise

16.9. GetFocal2

GetFocal2(ULONG* pdwFocal, ULONG* pdwPos, ULONG* bReturn)

Function	Get second focal parameters
pdwFocal	Current focal : 1 no focal 2 50 mm 3 250 mm
pdwPos	Current position
bReturn	TRUE if no error occured, FALSE otherwise

16.10. CanStopFocus

CanStopFocus (`ULONG* pbReturn`)

Function	Stop focal capability
`pbReturn`	TRUE if stop focus is available, FALSE otherwise

16.11. StopFocus

StopFocus (`ULONG* pbReturn`)

Function	Stop focal
`pbReturn`	TRUE if no error occurred, FALSE otherwise

16.12. CanChangeFocusSpeed

CanChangeFocusSpeed (ULONG* pbReturn)

Function	Change focus speed capability
pbReturn	TRUE if available, FALSE otherwise

16.13. SetFocusSpeed

SetFocusSpeed (ULONG dwFocal, ULONG dwPercent, ULONG* pbReturn)

Function	Change focus speed
dwFocal	Focal Id
dwPercent	Wanted speed percentage
pbReturn	TRUE if no error occurred, FALSE otherwise

16.14. IsFocalChangeAvailable

IsFocalChangeAvailable (ULONG* pbReturn)

Function	Change focal capabilty (only for multi-focal)
pbReturn	TRUE if available, FALSE otherwise

16.15. ChangeMultiFocal

ChangeMultiFocal (ULONG dwFocal, ULONG* pbReturn)

Function	Change focal
dwFocal	Wanted focal
pbReturn	TRUE if no error occurred, FALSE otherwise

17. Lockin Box functions

This section describes functions related to Vircam lockin box.

17.1. GetFrameUtilSize

GetFrameUtilSize(ULONG* pdwUtilPoint, ULONG* pdwUtilLine, LONG* plOffset, ULONG* bReturn)

Function	Get frame size without lockin information
pdwUtilPoint	Width
pdwUtilLine	Heigth
plOffset	Offset of the the image buffer where the first valid pixel is.
bReturn	TRUE if no error occured, FALSE otherwise

17.2. GetLockinVersion

GetLockinVersion(ULONG* pdwVersion)

Function	Get lockin version
pdwVersion	Current lockin version

17.3. SetLockinDelay

SetLockinDelay(ULONG dwDelay, ULONG* bReturn)

Function	Set frame delay (only with extrernal lockin)
dwDelay	Wanted frame delay
bReturn	TRUE if no error occured, FALSE otherwise

17.4. GetLockinDelay

GetLockinDelay (ULONG* pdwDelay, ULONG* pdwMaxDelay, ULONG* pbReturn)

Function	Get current frame delay (only with extrernal lockin)
pdwDelay	Current frame delay
pdwMaxDelay	Maximum frame delay
bReturn	TRUE if no error occured, FALSE otherwise

17.5. GetLockinThreshold

GetLockinThreshold (ULONG* pdwMaxThreshold, ULONG* pdwLowThreshold, ULONG* pdwHighThreshold, ULONG* pdwIdSignalDetected, ULONG* pbReturn)

Function	Get lockin threshold (available for internal lockin)
pdwMaxThreshold	Current maximum threshold
pdwLowThreshold	Current low threshold
pdwHighThreshold	Current high threshold
pdwIdSignalDetected	Current signal ID
pbReturn	TRUE if no error occured, FALSE otherwise

17.6. SetLockinThreshold

SetLockinThreshold (ULONG dwLowThreshold, ULONG dwHighThreshold, ULONG dwIdSignalDetected, ULONG* pbReturn)

Function	Set lockin threshold (available for internal lockin)
dwLowThreshold	Wanted low threshold
dwHighThreshold	Wanted high threshold
dwIdSignalDetected	Wanted signal Id
pbReturn	TRUE if no error occured, FALSE otherwise

17.7. SetLockinParam

SetLockinParam (ULONG dwN, ULONG dwPhase, ULONG* pbReturn)

Function	Set lockin parameters (available for internal lockin)
dwN	Wanted N factor
dwHighThreshold	Wanted phase
pbReturn	TRUE if no error occured, FALSE otherwise

17.8. GetLockinParam

GetLockinParam (ULONG* pdwN, ULONG* pdwPhase, ULONG* pbReturn)

Function	Get lockin parameters (available for internal lockin)
pdwN	Current N factor
dwHighThreshold	Current phase
pbReturn	TRUE if no error occured, FALSE otherwise

18. Trigger Out functions

This section describes functions related to Vircam Trigger Out.

18.1. GetTriggerOut

GetTriggerOut (ULONG* bEnableTriggerOut, ULONG* dwStartSourceType, ULONG* dwStartEdgeType, ULONG* dwStopSourceType, ULONG* dwStopEdgeType, ULONG* dwHigthStateTime, ULONG* bReturn)

Function	Get trigger output parameters
bEnableTriggerOut	TRUE if enable, FALSE otherwise
dwStartSourceType	Current start source type : 1 Timer 2 Top frame 3 Integration time 4 External trigger IN 5 Lockin min 6 Lockin max 7 Lockin min/max 8 Lockin multiple 1/N 9 Alarm
dwStartEdgeType	Current start edge type : 1 falling edge 2 rising edge
dwStopSourceType	Current stop source type : 1 Timer 2 Top frame 3 Integration time 4 External trigger IN 5 Lockin min 6 Lockin max 7 Lockin min/max 8 Lockin multiple 1/N 9 Alarm
dwStopEdgeType	Current stop edge type : 1 falling edge 2 rising edge
dwHigthStateTime	Not implemented
bReturn	TRUE if no error occured, FALSE otherwise

18.2. GetTriggerOutCapabilityEnum

GetTriggerOutCapabilityEnum (ULONG dwStartSourceType, ULONG dwStartEdgeType, ULONG dwIndexEnum, ULONG* dwStopSourceType, ULONG* dwStopEdgeType, ULONG* bReturn)

Function	Get trigger output capability
dwStartSourceType	Wanted start source type : 1 Timer 2 Top frame 3 Integration time 4 External trigger IN 5 Lockin min 6 Lockin max 7 Lockin min/max 8 Lockin multiple 1/N 9 Alarm
dwStartEdgeType	Wanted start edge type : 1 falling edge 2 rising edge
dwIndexEnum	Wanted enumeration index
dwStopSourceType	Current stop source type : 1 Timer 2 Top frame 3 Integration time 4 External trigger IN 5 Lockin min 6 Lockin max 7 Lockin min/max 8 Lockin multiple 1/N 9 Alarm
dwStopEdgeType	Current stop edge type : 1 falling edge 2 rising edge 3 both
bReturn	TRUE if no error occured, FALSE otherwise

18.3. SetTriggerOut

SetTriggerOut (ULONG bEnableTriggerOut, ULONG dwStartSourceType, ULONG dwStartEdgeType, ULONG dwStopSourceType, ULONG dwStopEdgeType, ULONG dwHigthStateTime, ULONG bReturn)

Function	Set trigger output parameters
bEnableTriggerOut	TRUE to enable, FALSE otherwise
dwStartSourceType	Wanted start source type : 1 Timer 2 Top frame 3 Integration time 4 External trigger IN 5 Lockin min 6 Lockin max 7 Lockin min/max 8 Lockin multiple 1/N 9 Alarm
dwStartEdgeType	Wanted start edge type : 1 falling edge 2 rising edge
dwStopSourceType	Wanted stop source type : 1 Timer 2 Top frame 3 Integration time 4 External trigger IN 5 Lockin min 6 Lockin max 7 Lockin min/max 8 Lockin multiple 1/N 9 Alarm
dwStopEdgeType	Wanted stop edge type : 1 falling edge 2 rising edge
dwHigthStateTime	Not implemented
bReturn	TRUE if no error occured, FALSE otherwise

19. Analog Video Output Automatic Gain functions

This section describes functions related to Automatic Gain Control applied to camera video output.

19.1. SetAutoGain

HRESULT SetAutoGain(ULONG bAuto, ULONG nFilter, ULONG* bReturn)

Function	Activate the AGC
bAuto	TRUE for activate Auto Gain, FALSE otherwise.
nFilter	Filter Index
bReturn	TRUE if no error occured, FALSE otherwise

19.2. GetAutoGain

HRESULT GetAutoGain(ULONG* bAuto, FLOAT* fThreshold, LONG* nMethod, FLOAT* fDelta, LONG* nFilter, ULONG* bReturn)

Function	Retreive AGC state and parameters
bAuto	TRUE if Auto Gain is activated
fThreshold	Current threshold
nMethod	AGC method
fDelta	Current Delta
nFilter	Current filter index
bReturn	TRUE if no error occured, FALSE otherwise

19.3. SetAGCPercentage

HRESULT SetAGCPercentage(FLOAT fPercentage, ULONG* bReturn)

Function	Set AGC percentage
fPercentage	Percentage
bReturn	TRUE if no error occured, FALSE otherwise

19.4. GetAGCPercentage

HRESULT GetAGCPercentage(FLOAT* fPercentage, ULONG* bReturn)

Function	Get AGC percentage
fPercentage	Percentage
bReturn	TRUE if no error occured, FALSE otherwise

19.5. SetAGCHysteresis

HRESULT SetAGCHysteresis(FLOAT fHysteresis, ULONG* bReturn)

Function	Set AGC Hysteresis
fHysteresis	Hysteresis
bReturn	TRUE if no error occured, FALSE otherwise

19.6. GetAGCHysteresis

HRESULT GetAGCHysteresis(FLOAT* fHysteresis, ULONG* bReturn)

Function	Get AGC Hysteresis
fHysteresis	Hysteresis
bReturn	TRUE if no error occured, FALSE otherwise

19.7. SetAGCLimit

HRESULT SetAGCLimit(ULONG bLimit, FLOAT fGainMax, FLOAT fGainMin, ULONG* bReturn)

Function	Set AGC Limits
bLimit	TRUE for activate AGC Limits, FALSE otherwise
fGainMax	Maximum Gain
fGainMin	Minimum Gain
bReturn	TRUE if no error occured, FALSE otherwise

19.8. GetAGCLimit

HRESULT GetAGCLimit(ULONG* bLimit, FLOAT* fGainMax, FLOAT* fGainMin, ULONG* bReturn)

Function	Get AGC Limits
bLimit	TRUE if AGC Limits is activated, FALSE otherwise
fGainMax	Maximum Gain
fGainMin	Minimum Gain
bReturn	TRUE if no error occured, FALSE otherwise

19.9. SetAGCSmooth

HRESULT SetAGCSmooth(ULONG bActive, FLOAT fCoef1, FLOAT fCoef2, FLOAT fCoef3, FLOAT fCoef4, ULONG* bReturn)

Function	Set AGC Smoothing parameters
bActive	Enable / Disable Smoothing (TRUE/FALSE)
fCoef1	Coefficient 1
fCoef2	Coefficient 2
fCoef3	Coefficient 3
fCoef4	Coefficient 4
bReturn	TRUE if no error occured, FALSE otherwise

19.10. GetAGCSmooth

HRESULT GetAGCSmooth(ULONG* bActive, FLOAT* fCoef1, FLOAT* fCoef2, FLOAT* fCoef3, FLOAT* fCoef4, ULONG* bReturn)

Function	Get AGC Smoothing parameters
bActive	Smoothing state (TRUE/FALSE)
fCoef1	Current Coefficient 1
fCoef2	Current Coefficient 2
fCoef3	Current Coefficient 3
fCoef4	Current Coefficient 4
bReturn	TRUE if no error occured, FALSE otherwise

19.11. SetOffsetLimit

HRESULT SetOffsetLimit(ULONG bActive, ULONG wOffsetMin, ULONG wOffsetMax, ULONG* bReturn)

Function	Set offset limit parameters
bActive	TRUE to Enable offset limit, FALSE otherwise
wOffsetMin	Wanted minimum offset
wOffsetMax	Wanted maximum offset
bReturn	TRUE if no error occured, FALSE otherwise

19.12. GetOffsetLimit

HRESULT GetOffsetLimit(ULONG* bActive, ULONG* wOffsetMin, ULONG* wOffsetMax, ULONG* bReturn)

Function	Get offset limits parameters
bActive	TRUE if offset limit is enable, FALSE otherwise
wOffsetMin	Current minimum offset
wOffsetMax	Current maximum offset
bReturn	TRUE if no error occured, FALSE otherwise

19.13. SetGammaCorrection

HRESULT SetGammaCorrection(ULONG bActive, FLOAT fGamma, ULONG* bReturn)

Function	Set gamma correction
bActive	TRUE to enable gamma correction, FALSE otherwise
Parameter 2	Wanted gamma value
bReturn	TRUE if no error occured, FALSE otherwise

19.14. GetGammaCorrection

HRESULT GetGammaCorrection(ULONG* bActive, FLOAT* fGamma, ULONG* bReturn)

Function	Récupérer les Parameters de correction gamma
bActive	TRUE if gamma correction is enabled, FALSE otherwise
fGamma	Current gamma value
bReturn	TRUE if no error occured, FALSE otherwise

19.15. SetRoi

HRESULT SetRoi(ULONG bRoi, ULONG wWidth, ULONG wHeigth, ULONG wLeft, ULONG wHigth, ULONG* bReturn)

Function	Set Region of interest parameters
bRoi	TRUE to enable it, FALSE otherwise
wWidth	Wanted width
wHeigth	Wanted heigth
wLeft	Wanted left
wHigth	Wanted top
bReturn	TRUE if no error occured, FALSE otherwise

19.16. GetRoi

HRESULT GetRoi(ULONG* bRoi, ULONG* wWidth, ULONG* wHeigth, ULONG* wLeft, ULONG* wHigth, ULONG* bReturn)

Function	Get Région of interest parameters
bRoi	TRUE if it is enable, FALSE otherwise
wWidth	Current width
wHeigth	Current heigth
wLeft	Current left
wHigth	Current top
bReturn	TRUE if no error occured, FALSE otherwise

19.17. SetGain

HRESULT SetGain(FLOAT fGain, ULONG* bReturn)

Function	Set video gain
fGain	Wanted gain
bReturn	TRUE if no error occured, FALSE otherwise

19.18. SetOffset

HRESULT SetOffset(ULONG wOffset, ULONG* bReturn)

Function	Set video offset
wOffset	Wanted offset
bReturn	TRUE if no error occured, FALSE otherwise

19.19. GetOffsetGain

HRESULT GetOffsetGain(ULONG * pdwOffset, FLOAT* pfGain, ULONG* bReturn)

Function	Get video gain and offset
pdwOffset	Current video offset
pfGain	Current video Gain
bReturn	TRUE if no error occured, FALSE otherwise

20. Video tools functions

This section describes functions related to camera video output tools.

20.1. GetZoom

HRESULT GetZoom(ULONG* bZoom, LONG* nFilter, ULONG* bReturn)

Function	Get zoom state
bZoom	TRUE if video Zoom is active, FALSE otherwise
nFilter	Current filter number
bReturn	TRUE if no error occured, FALSE otherwise

20.2. SetZoom

HRESULT SetZoom(ULONG bZoom, LONG nFilter, ULONG* bReturn)

Function	Set zoom state
bZoom	Activate/ Déactivate zoom (TRUE/FALSE)
nFilter	Wanted filter number
bReturn	TRUE if no error occured, FALSE otherwise

20.3. CanChangePalette

HRESULT CanChangePalette(ULONG* bReturn)

Function	Can change video palette
bReturn	TRUE if change palette is available, FALSE otherwise

20.4. SetPalette

HRESULT SetPalette(ULONG bSwapp, ULONG dwPalette, ULONG* bReturn)

Function	Set palette parameters
bSwapp	TRUE to swapp palette, FALSE otherwise
dwPalette	Wanted palette number
bReturn	TRUE if no error occured, FALSE otherwise

20.5. GetPalette

HRESULT GetPalette(ULONG* bSwapp, ULONG* pdwPalette, ULONG* bReturn)

Function	Get palette parameters
bSwapp	TRUE if palette is swapped, FALSE otherwise
pdwPalette	Current palette number
bReturn	TRUE if no error occured, FALSE otherwise

20.6. SetReticle

HRESULT SetReticle(ULONG bReticle, ULONG* bReturn)

Function	Set reticle state
bReticle	Set TRUE to show reticle, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

20.7. GetReticle

HRESULT GetReticle(ULONG* pbReticle, ULONG* bReturn)

Function	Get reticle state
pbReticle	TRUE if reticle is enable, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

20.8. SetVideoDisable

HRESULT SetVideoDisable(ULONG bDisable, ULONG bAutoFrequency, ULONG* bReturn)

Function	Enable or disable video output
bDisable	TRUE to disable video, FALSE otherwise
bAutoFrequency	TRUE to change automatically the frame rate when necessary, FALSE otherwise. Example: When the Frequency is too important for the future windowing.
bReturn	TRUE if no error occured, FALSE otherwise

20.9. GetVideoDisable

HRESULT GetVideoDisable(ULONG* pbDisable, ULONG* bReturn)

Function	Get video output state
pbDisable	If TRUE video is disable, otherwise FALSE
bReturn	TRUE if no error occured, FALSE otherwise

20.10. SetVideoType

HRESULT SetVideoType(ULONG dwVideo, ULONG* bReturn)

Function	Set video type.
dwVideo	Wanted video type : 50 (CCIR) 60 (RS170)
bReturn	TRUE if no error occured, FALSE otherwise

20.11. GetVideoType

HRESULT GetVideoType(ULONG* pdwVideo, ULONG* bReturn)

Function	Get Video type
pdwVideo	Current video type : 50 (CCIR) 60 (RS170)
bReturn	TRUE if no error occured, FALSE otherwise

20.12. FlipH

HRESULT FlipH(ULONG* bReturn)

Function	Flip image horizontaly
bReturn	TRUE if no error occured, FALSE otherwise

20.13. FlipV

HRESULT FlipV(ULONG* bReturn)

Function	Flip image verticaly
bReturn	TRUE if no error occured, FALSE otherwise

20.14. IsFlipHAvailable

HRESULT IsFlipHAvailable(ULONG* bReturn)

Function	Horizontal Flip Image available
bReturn	TRUE if available, FALSE otherwise

20.15. IsFlipVAvailable

HRESULT IsFlipVAvailable(ULONG* bReturn)

Function	Vertical Flip Image available
bReturn	TRUE if no error occured, FALSE otherwise

20.16. SetLiveImage

SetLiveImage(ULONG bLive, ULONG* bReturn)

Function	Set live video output
bLive	TRUE if video is not freezed, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

20.17. CanFreeze

HRESULT CanFreeze(ULONG* bReturn)

Function	Freeze available
bReturn	TRUE if video can be freezed, FALSE otherwise

20.18. IsFreeze

HRESULT IsFreeze(ULONG* bReturn)

Function	Video output live state
bReturn	TRUE if video output is freezed, FALSE otherwise

20.19. SetLut

HRESULT SetLut(ULONG wMin, ULONG wMax, ULONG* bReturn)

Function	Set LUT parameters
wMin	Wanted minimum level
wMax	Wanted maximum level
bReturn	TRUE if no error occured, FALSE otherwise

20.20. SetMap

HRESULT SetMap(ULONG bMap, ULONG* bReturn)

Function	Activate mapping
bMap	TRUE to enable mapping, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

20.21. GetMap

HRESULT GetMap(ULONG* pbMap, LONG* pnAngle, LONG* pnFlip, ULONG* bReturn)

Function	Get mapping parameters
pbMap	TRUE if mapping is enable, FALSE otherwise
pnAngle	Current rotations angles
pnFlip	Flip : 0 no flip 1 vertical flip 2 horizontal flip
bReturn	TRUE if no error occured, FALSE otherwise

20.22. SetRotation

HRESULT SetRotation(LONG nAngle, ULONG* bReturn)

Function	Set rotations angles
nAngle	Wanted angles
bReturn	TRUE if no error occured, FALSE otherwise

20.23. GetImageInformation

HRESULT GetImageInformation(ULONG* pdwMin, ULONG* pdwMax, ULONG* pdwMean, FLOAT* pfStd, ULONG* bReturn)

Function	Get image information
pdwMin	Minimum digital level
pdwMax	Maximum digital level
pdwMean	Mean digital level
pfStd	pfstd variant
bReturn	TRUE if no error occured, FALSE otherwise

20.24. IsImageInformationAvailable

HRESULT IsImageInformationAvailable(ULONG* pbAvailable, ULONG* bReturn)

Function	Is image information available
pbAvailable	TRUE when you can use « GetImageInformation », FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

20.25. GetLowRes

HRESULT GetLowRes(ULONG* pbLowRes, ULONG* bReturn)

Function	Get low resolution state
Parameter 1	TRUE if low resolution is enable, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

20.26. SetLowRes

HRESULT SetLowRes(ULONG bLowRes, ULONG* bReturn)

Function	Set low resolution state
bLowRes	TRUE to be in low resolution, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

20.27. SetCompensation

HRESULT SetCompensation(ULONG bCompensation, FLOAT fK1, FLOAT fK2, FLOAT fK3, ULONG* bReturn)

Function	Get compensation parameters
bCompensation	TRUE to enable compensation
fK1	Constant degree Polynomial parameter
fK2	First degree Polynomial parameter
fK3	Second degree Polynomial parameter
bReturn	TRUE if no error occured, FALSE otherwise

20.28. GetCompensation

HRESULT GetCompensation(ULONG* pbCompensation, FLOAT* pfK1, FLOAT* pfK2, FLOAT* pfK3, ULONG* bReturn)

Function	Get compenstion parameters
pbCompensation	TRUE if compensation is enable, FALSE otherwise
pfK1	Constant degree Polynomial parameter
pfK2	First degree Polynomial parameter
pfK3	Second degree Polynomial parameter
bReturn	TRUE if no error occured, FALSE otherwise

20.29. GetContinuousZoom

HRESULT GetContinuousZoom(FLOAT* pfMultiplicator, ULONG* pbReturn)

Function	Get Video continuous zoom value (only for IRV300&IRV50)
pfMultiplicator	Zoom multiplicator
pbReturn	TRUE if no error occured, FALSE otherwise

20.30. SetContinuousZoom

HRESULT SetContinuousZoom(FLOAT fMultiplicator, ULONG* pbReturn)

Function	Set Video continuous zoom value (only for IRV300&IRV50)
fMultiplicator	Zoom multiplicator
pbReturn	TRUE if no error occured, FALSE otherwise

21. NUC & BPR functions

This section describes functions related to camera non uniformity correction and bad pixel representation.

21.1. SetCurrentNuc

HRESULT SetCurrentNuc(LONG nNuc, ULONG* bReturn)

Function	Select NUC
nNuc	Wanted NUC index
bReturn	TRUE if no error occured, FALSE otherwise

21.2. GetCurrentNUC

HRESULT GetCurrentNUC(LONG* pnNuc, ULONG* bReturn)

Function	Get current NUC index
pnNuc	Current « NUC » index
bReturn	TRUE if no error occured, FALSE otherwise

21.3. GetNbrNUC

HRESULT GetNbrNUC(LONG* pnNbrNuc, ULONG* bReturn)

Function	Get number of NUC
pnNbrNuc	Current number of NUC
bReturn	TRUE if no error occured, FALSE otherwise

21.4. SaveNucConfig

HRESULT SaveNucConfig(LONG nNuc, ULONG* bReturn)

Function	Save NUC
nNuc	NUC index that represent on witch NUC it will be saved
bReturn	TRUE if no error occured, FALSE otherwise

21.5. DownloadNuc

HRESULT DownloadNuc(BSTR strOutputFile, ULONG* bReturn)

Function	Download NUC from camera to PC
strOutputFile	Basic string (BSTR) that represent the output file
bReturn	TRUE if no error occured, FALSE otherwise

21.6. UploadNuc

HRESULT UploadNuc(BSTR strInputFile, ULONG* bReturn)

Function	Upload Nuc from PC to camera
strInputFile	Basic string (BSTR) that represent the file witch will be upload in the camera
bReturn	TRUE if no error occured, FALSE otherwise

21.7. DownloadBPR

HRESULT DownloadBPR(BSTR strOutputFile, ULONG* bReturn)

Function	Download BPR from Camera to PC
strOutputFile	Basic string (BSTR) that represents file destination
bReturn	TRUE if no error occured, FALSE otherwise

21.8. UploadBpr

HRESULT UploadBpr(BSTR strInputFile, ULONG* bReturn)

Function	Upload from PC to Camera
strInputFile	Basic string (BSTR) that represents source file
bReturn	TRUE if no error occured, FALSE otherwise

21.9. GetCamTemp

GetCamTemp(int nIndex, FLOAT* pfTemp, ULONG* bReturn)

Function	Get camera temperature by sensor
nIndex	Sensor index
pfTemp	Sensor temperature
bReturn	TRUE if no error occured, FALSE otherwise

21.10. UploadDataNucComp

UploadDataNucComp(BSTR strInputFile, ULONG* pbReturn)

Function	Upload CNUC™ from PC to camera
strInputFile	Basic string (BSTR) that represent the file witch will be upload in the camera
pbReturn	TRUE if no error occured, FALSE otherwise

21.11. DownloadDataNucComp

DownloadDataNucComp(BSTR strInputFile, ULONG* pbReturn)

Function	Download CNUC™ from camera to PC
strInputFile	Basic string (BSTR) that represent the output file
pbReturn	TRUE if no error occured, FALSE otherwise

21.12. GetNUCPageNumber

GetNUCPageNumber(LONG* pnPageNumber, ULONG* pbReturn)

Function	Get number of NUC page
pnPageNumber	Current number of NUC page used by camera
pbReturn	TRUE if no error occured, FALSE otherwise

21.13. GetNUCFileId

GetNucFileId(LONG nTableIndex, LONG* pnFileId,ULONG* pbReturn)

Function	Get file identifier for a choosed Nuc index
nTableIndex	Wanted Nuc index
pnFileId	Nuc file identifier
pbReturn	TRUE if no error occured, FALSE otherwise

21.14. UploadTri

UploadTri(BSTR strInputFile, ULONG* pbReturn)

Function	Upload Tri-Lut
strInputFile	Basic string (BSTR) that represent the file witch will be upload in the camera
pbReturn	TRUE if no error occured, FALSE otherwise

22. NUC process functions

This section describes functions related to camera non uniformity correction process.

22.1. CalculateNuc

CalculateNuc (ULONG* bReturn)

Function	Calculate NUC
bReturn	TRUE if no error occured, FALSE otherwise

22.2. SetMethode

HRESULT SetMethode(LONG nMethode, ULONG* bReturn)

Function	Set NUC calculation method
nMethode	0 Integration time method +/- 5% 1 Integration time method by specified IT 2 Black body method 3 Shutter method (only for one point calculation method)
bReturn	TRUE if no error occured, FALSE otherwise

22.3. GetMethod

HRESULT GetMethod(LONG* pnMethod, ULONG* bReturn)

Function	Get NUC calculation method
pnMethod	Current used method
bReturn	TRUE if no error occured, FALSE otherwise

22.4. GetNucType

HRESULT GetNucType(LONG* pnType, ULONG* bReturn)

Function	Get NUC calculation type (1 point/2 point)
pnType	Type : 0 Two points calculation 1 One point calculation
bReturn	TRUE if no error occured, FALSE otherwise

22.5. SetNucType

HRESULT SetNucType(LONG nType, ULONG* bReturn)

Function	Set NUC calculation type (1 point/2 point)
nType	Type : 0 Two points calculation 1 One point calculation
bReturn	TRUE if no error occured, FALSE otherwise

22.6. SetDoNuc

HRESULT SetDoNuc(ULONG bDoNuc, ULONG* bReturn)

Function	Set NUC calculation option
bDoNuc	Choose TRUE and NUC will be done, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

22.7. UpdateNuc

HRESULT UpdateNuc(ULONG* bReturn)

Function	Update NUC configuration with current TI, Frame rate, Filter
bReturn	TRUE if no error occured, FALSE otherwise

22.8. GetThreshold1

HRESULT GetThreshold1(LONG* pnThresold, ULONG* bReturn)

Function	Get low Threshold for first method (1)
pnThresold	Current Integration time
bReturn	TRUE if no error occured, FALSE otherwise

22.9. SetThreshold1

HRESULT SetThreshold1(LONG nThreshold, ULONG* bReturn)

Function	Set low Threshold for first method (1)
nThreshold	Wanted Integration time
bReturn	TRUE if no error occured, FALSE otherwise

22.10. GetThreshold2

HRESULT GetThreshold2(LONG* pnThresold, ULONG* bReturn)

Function	Get higth Threshold for first method (1)
pnThresold	Current Integration time
bReturn	TRUE if no error occured, FALSE otherwise

22.11. SetThreshold2

HRESULT SetThreshold2(LONG nThreshold, ULONG* bReturn)

Function	Set higth Threshold for first method (1)
nThreshold	Wanted Integration time
bReturn	TRUE if no error occured, FALSE otherwise

22.12. GetAverageFrames

HRESULT GetAverageFrames(LONG* pnFrames, ULONG* bReturn)

Function	Get number of frame used to calculate NUC average
pnFrames	Current number of frame
bReturn	TRUE if no error occured, FALSE otherwise

22.13. SetAverageFrames

HRESULT SetAverageFrames(LONG nFrames, ULONG* bReturn)

Function	Set number of frame used to calculate NUC average
nFrames	Wanted number of frame
bReturn	TRUE if no error occured, FALSE otherwise

22.14. GetGainKeep

HRESULT GetGainKeep(ULONG* pbKeepGain, ULONG* bReturn)

Function	Get Keep Gain option for NUC calculation state
pbKeepGain	TRUE if gain is kept, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

22.15. SetKeepGain

HRESULT SetKeepGain(ULONG bKeepGain, ULONG* bReturn)

Function	Set Keep Gain option for NUC calculation state
bKeepGain	TRUE to keep gain, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

22.16. IsWarningNuc

HRESULT IsWarningNuc(ULONG* pbNUCWarning, ULONG* bReturn)

Function	Get warning NUC option state.
pbNUCWarning	If TRUE, when you calculate a new NUC if the last one was not saved a warning window appeared
bReturn	TRUE if no error occured, FALSE otherwise

22.17. SetWarningNuc

HRESULT SetWarningNuc(ULONG bNUCWarning, ULONG* bReturn)

Function	Change warning NUC option state.
bNUCWarning	TRUE to enable warning NUC option, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

22.18. IsUpdatedButNotSaved

HRESULT IsUpdatedButNotSaved(ULONG* pbUpdatedButNotSaved, ULONG* bReturn)

Function	Is NUC calculation save state
pbUpdatedButNotSaved	TRUE if last NUC calculation was not saved, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

22.19. SetUpdatedButNotSaved

HRESULT SetUpdatedButNotSaved(ULONG bUpdatedButNotSaved, ULONG* bReturn)

Function	Set NUC calculation save state
bUpdatedButNotSaved	TRUE to force to not saved, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

22.20. IsNucToDo

HRESULT IsNucToDo(ULONG* pbDoNuc, ULONG* bReturn)

Function	NUC calculation option
pbDoNuc	TRUE if NUC calculation have to be done in NUC and BPR process, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

22.21. GetSaveAfterUpdate

HRESULT GetSaveAfterUpdate(ULONG* pbSaveAfterUpdate, ULONG* bReturn)

Function	Get automatically save after update option
pbSaveAfterUpdate	TRUE if enable, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

22.22. SetSaveAfterUpdate

HRESULT SetSaveAfterUpdate(ULONG bSaveAfterUpdate, ULONG* bReturn)

Function	Set automatically save after update option
bSaveAfterUpdate	TRUE to save NUC in flash after calculation, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

22.23. NUC Calculation procedure

To process NUC, first you have to choose calculation method and type as you want with:

- SetMethode
- SetAverageFrames
- SetNucType
- SetDoNuc

Depending on the method or type you used you have to configure it with:

- SetThreshold1
- SetThres hold2
- SetKeepGain

At last you can after that process NUC with:

- CalculateNuc

If you directly process NUC Virtual Cam will use default values if it is the first time, last values otherwise.

22.24. GetNucLensName

HRESULT GetNucLensName(LONG nTableIndex, BSTR* pstrLensName, ULONG* pbReturn)

Function	Get NUC lens Name
nTableIndex	NUC table index
pstrLensName	Lens Name
bReturn	TRUE if no error occured, FALSE otherwise

22.25. GetNucFilterName

HRESULT `GetNucFilterName(LONG nTableIndex, BSTR* pstrFilterName, ULONG* pbReturn)`

Function	Get NUC Filter Name
nTableIndex	NUC table index
pstrLensName	Filter Name
bReturn	TRUE if no error occured, FALSE otherwise

23. BPR process functions

This section describes functions related to camera bad pixel process.

23.1. CalculateBPR

HRESULT CalculateBPR(ULONG* bReturn)

Function	Calculate « BPR »
bReturn	TRUE if no error occured, FALSE otherwise

23.2. GetUpdateOrResetBPRList

HRESULT GetUpdateOrResetBPRList(LONG* pnStatus, ULONG* bReturn)

Function	Get state of update or reset BPR list option
pnStatus	Status : 0 Reset 1 Update
bReturn	TRUE if no error occured, FALSE otherwise

23.3. SetUpdateOrResetBPRList

HRESULT SetUpdateOrResetBPRList(LONG nStatus, ULONG* bReturn)

Function	Change state of update or reset BPR list option
nStatus	Status : 0 Reset 1 Update
bReturn	TRUE if no error occured, FALSE otherwise

23.4. SetUseNoisyMethode

HRESULT SetUseNoisyMethode(ULONG bNoisy, ULONG* bReturn)

Function	Use noisy method
bNoisy	TRUE to select noisy method, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

23.5. IsNoisyMethodeChoosed

HRESULT IsNoisyMethodeChoosed(ULONG* pbNoisy, ULONG* bReturn)

Function	Get noisy method activated
pbNoisy	TRUE if noisy method is to be used in process, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

23.6. SetBPRNoise

HRESULT SetBPRNoise(FLOAT fNoise, ULONG* bReturn)

Function	Set noise threshold
fNoise	Wanted noisy threshold
bReturn	TRUE if no error occured, FALSE otherwise

23.7. SetEnableBPRCalculation

HRESULT SetEnableBPRCalculation(ULONG bEnableBPRCalculation, ULONG* bReturn)

Function	Set enable BPR calculation
bEnableBPRCalculation	TRUE to enable BPR calculation, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

23.8. IsEnableBPRCalculation

HRESULT IsEnableBPRCalculation(ULONG* pbEnableBPRCalculation, ULONG* bReturn)

Function	Is enable BPR in process
pbEnableBPRCalculation	TRUE if BPR calculation is enable in the process, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

23.9. GetBPRNoise

HRESULT GetBPRNoise(FLOAT* pfNoise, ULONG* bReturn)

Function	Get noise threshold
pfNoise	Current noisy threshold
bReturn	TRUE if no error occured, FALSE otherwise

23.10. SetUseResponsivityMethode

HRESULT SetUseResponsivityMethode(ULONG bResponsivity, ULONG* bReturn)

Function	Use responsivity method
bResponsivity	TRUE to select responsivity method, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

23.11. IsResponsivityMethodeChoosed

HRESULT IsResponsivityMethodeChoosed(ULONG* pbResponsivity, ULONG* bReturn)

Function	Get noisy method activated
pbResponsivity	TRUE if responsivity method is to be used in process, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

23.12. SetBPRResponseLevel

HRESULT SetBPRResponseLevel(LONG nResponseLevel, ULONG* bReturn)

Function	Set response threshold
nResponseLevel	Wanted response threshold
bReturn	TRUE if no error occured, FALSE otherwise

23.13. GetBPRResponseLevel

HRESULT GetBPRResponseLevel(LONG* pnResponseLevel, ULONG* bReturn)

Function	Get response threshold
pnResponseLevel	Current response threshold
bReturn	TRUE if no error occured, FALSE otherwise

23.14. SetUseOffsetMethode

HRESULT SetUseOffsetMethode(ULONG bOffset, ULONG* bReturn)

Function	Use offset method
bOffset	TRUE to select offset method
bReturn	TRUE if no error occured, FALSE otherwise

23.15. IsOffsetMethodeChoosed

HRESULT IsOffsetMethodeChoosed(ULONG* pbOffset, ULONG* bReturn)

Function	Get response method activated
pbOffset	TRUE if ofsset method is selected, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

23.16. SetBPRDelta

HRESULT SetBPRDelta(LONG nDelta, ULONG* bReturn)

Function	Set BPR offset
nDelta	Wanted offset threshold
bReturn	TRUE if no error occured, FALSE otherwise

23.17. GetBPRDelta

HRESULT GetBPRDelta(LONG* pnDelta, ULONG* bReturn)

Function	Get BPR offset
pnDelta	Current offset threshold
bReturn	TRUE if no error occured, FALSE otherwise

23.18. SetBPRAccumulateFrame

HRESULT SetBPRAccumulateFrame(ULONG dwFrame, ULONG* bReturn)

Function	Set number of frames for BPR accumulation process
dwFrame	Wanted number of frames
bReturn	TRUE if no error occured, FALSE otherwise

23.19. GetBPRAccumulateFrame

HRESULT GetBPRAccumulateFrame(ULONG* pdwFrames, ULONG* bReturn)

Function	Get number of frames for BPR accumulation process
pdwFrames	Current number of frames
bReturn	TRUE if no error occured, FALSE otherwise

23.20. GetBPRuseShutter

HRESULT GetBPRuseShutter (ULONG* pbReturn)

Function	Get number of frames for BPR accumulation process (only for noisy method)
pbReturn	TRUE if used, FALSE otherwise

23.21. SetBPRuseShutter

HRESULT SetBPRuseShutter (ULONG bUseShutter, ULONG* pbReturn)

Function	Get number of frames for BPR accumulation process (only for noisy method)
bUseShutter	TRUE to use shutter, FALSE otherwise
pbReturn	TRUE if used, FALSE otherwise

23.22. BPR calculation procedure

First you have too choosed witch method you want to use to process with BPR calculation, if you want to update the existing BPR list or to replace it by new.

Using differents method like:

- SetUseNoisyMethode
- SetUseResponsivityMethode
- SetUseOffsetMethode
- SetUpdateOrResetBPRList

Then depending on the method used you would have to set parameters with:

- SetBPRAccumulatedFrame
- SetBPRResponseLevel
- SetBPRNoise
- SetBPRDelta

At last you can after that process BPR with:

- CalculateBPR

24. NUC Configuration functions

This section describes functions related to Vircam NUC configuration.

24.1. GetConfigTi

HRESULT GetConfigTi(LONG nConfig, ULONG* pdwTi, ULONG* bReturn)

Function	Get integration time for a NUC configuration
nConfig	NUC configuration index
pdwTi	Current configuration Integration time
bReturn	TRUE if no error occurred, FALSE otherwise

24.2. GetConfigFrequency

HRESULT GetConfigFrequency(ULONG dwConfig, FLOAT* pfFrequency, ULONG* bReturn)

Function	Get frame rate by NUC configuration
dwConfig	NUC configuration index
pfFrequency	Current configuration frame rate
bReturn	TRUE if no error occured, FALSE otherwise

24.3. GetConfigStringInfo

HRESULT GetConfigStringInfo(LONG nConfig, BSTR* pStrConfig, ULONG* bReturn)

Function	Get a string that represents the NUC config
nConfig	NUC configuration index
pStrConfig	(BSTR : Basic String) Current string configuration
bReturn	TRUE if no error occured, FALSE otherwise

24.4. SetConfigAutoChange

HRESULT SetConfigAutoChange(ULONG bSet, ULONG* bReturn)

Function	Set synchronize on NUC configuration
bSet	TRUE to synchronize integration time, frame rate, filter on NUC configuration
bReturn	TRUE if no error occured, FALSE otherwise

24.5. IsAutoChangeConfig

HRESULT IsAutoChangeConfig(ULONG* bReturn)

Function	Get synchronize on NUC configuration state
bReturn	TRUE if integration time, frame rate and filter are synchronized on NUC configuration

24.6. GetConfigName

HRESULT GetConfigName(ULONG dwConfig, BSTR* pstrConfigName, ULONG* bReturn)

Function	Get NUC configuration name
dwConfig	Wanted NUC configuration index
pstrConfigName	(BSTR : Basic String) Configuration name
bReturn	TRUE if no error occured, FALSE otherwise

24.7. SetConfigName

HRESULT SetConfigName(ULONG dwConfig, BSTR strConfigName, ULONG* bReturn)

Function	Set NUC configuration name
dwConfig	Wanted NUC configuration index
strConfigName	(BSTR : Basic String) Wanted configuration name
bReturn	TRUE if no error occured, FALSE otherwise

24.8. GetConfig

HRESULT GetConfig(ULONG* bReturn)

Function	Get all camera current configuration
bReturn	TRUE if no error occured, FALSE otherwise

25. Detector functions

This section describes functions related to detector driving.

- WARNING THIS MUST BE USED WITH GREAT CARE!
- DO NOT MAKE A CALL TO ONE ONE THIS FUNCTIONS UNLESS YOU ARE PERFECTLY AWARE OF THE CONSEQUENCES.
- CAMERA CAN BE DAMAGED DUE TO A WRONG USE OF THESE FUNCTIONS.

25.1. SetExternal

HRESULT SetExternal(ULONG `dwExternal`, ULONG* bReturn)

Function	Set external trigger
`dwExternal`	0 External synchro desactivated. 1 External trigger 4 Lockin synchro (only for internal lockin)
bReturn	TRUE if no error occured, FALSE otherwise

25.2. GetExternal

HRESULT GetExternal(ULONG* `pdwExternal`, ULONG* bReturn)

Function	Get external trigger
`pdwExternal`	0 External synchro desactivated. 1 External trigger 4 Lockin synchro (only for internal lockin)
bReturn	TRUE if no error occured, FALSE otherwise

25.3. SetGenlock

HRESULT SetGenlock(ULONG bGenlock, ULONG* bReturn)

Function	Set GenLock state option
bGenlock	TRUE to activate external synchronization, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

25.4. GetGenlock

HRESULT GetGenlock(ULONG* pbGenLock, ULONG* bReturn)

Function	Get GenLock state option
pbGenLock	TRUE if external synchronization is activated, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

25.5. GetSensorTemperature

HRESULT GetSensorTemperature(ULONG* pdwTemperature, ULONG* bReturn)

Function	Get detector temperature
pdwTemperature	Current temperature
bReturn	TRUE if no error occured, FALSE otherwise

25.6. GetDetector

HRESULT GetDetector(LONG* plDetector, ULONG* bReturn)

Function	Get detector code
plDetector	Detector code
bReturn	TRUE if no error occured, FALSE otherwise

25.7. IsDetector

HRESULT IsDetector(LONG lDetector, ULONG* pbTheOne, ULONG* bReturn)

Function	Test a detector code compared to camera's detector code
lDetector	Detector code to test
pbTheOne	TRUE if the detector code to test is the very camera detector code, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

25.8. GetDetectorType

HRESULT GetDetectorType(ULONG* pdwDetector, ULONG* pdwPoint, ULONG* pdwLine, ULONG* pdwBit, ULONG* pdwDetector2, ULONG* bReturn)

Function	Get detector parameters
pdwDetector	First part of detector code (Least Significant Word)
pdwPoint	Current detector width
pdwLine	Current detector heigth
pdwBit	Number of bit per pixel
pdwDetector2	Second part of detector code (Most Significant Word)
bReturn	TRUE if no error occured, FALSE otherwise

25.9. GetMasterClock

HRESULT GetMasterClock(FLOAT* pfOscillator, FLOAT* pfMasterClock, ULONG* bReturn)

Function	Get master clock
pfOscillator	Current oscillator
pfMasterClock	Current master clock
bReturn	TRUE if no error occured, FALSE otherwise

25.10. SetMasterClock

HRESULT SetMasterClock(FLOAT fOscillator, FLOAT fMasterClock, ULONG* bReturn)

Function	Set master clock
fOscillator	Wanted oscillator
fMasterClock	Wanted master clock
bReturn	TRUE if no error occured, FALSE otherwise

25.11. GetSkimming

HRESULT GetSkimming(ULONG* pbSkimming, LONG* pnCurrent, LONG* pnBias, LONG* pnPower, LONG* pnColumnPower, ULONG* bReturn)

Function	Get skimming parameters
pbSkimming	TRUE if skimming is activated, FALSE otherwise
pnCurrent	Current skimming
pnBias	Bias skimming
pnPower	Power skimming
pnColumnPower	Current power column skimming
bReturn	TRUE if no error occured, FALSE otherwise

25.12. SetSkimming

HRESULT SetSkimming(ULONG bSkimming, LONG nCurrent, LONG nBias, LONG nPower, LONG nColumnPower, ULONG* bReturn)

Function	Set skimming parameters
bSkimming	TRUE to activate skimming, FALSE otherwise
nCurrent	Current skimming
nBias	Bias skimming
nPower	Power skimming
nColumnPower	Current power column skimming
bReturn	TRUE if no error occured, FALSE otherwise

25.13. IsTTLAllowed

HRESULT IsTTLAllowed(ULONG* pbAllowed, ULONG* bReturn)

Function	Is External Triggering implemented
pbAllowed	TRUE if TTL is allowed, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

25.14. SetTTLAllowed

HRESULT SetTTLAllowed(ULONG bAllow, ULONG* bReturn)

Function	Set External Triggering implemented
bAllow	TRUE to allow TTL, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

25.15. IsGenLockAllowed

HRESULT IsGenLockAllowed(ULONG* pbAllowed, ULONG* bReturn)

Function	Is Genlock implemented
pbAllowed	TRUE if Genlock is allowed, FALSE otherwise
bReturn	TRUE if no error occurred, FALSE otherwise

25.16. SetGenLockAllowed

HRESULT SetGenLockAllowed(ULONG bAllow, ULONG* bReturn)

Function	Set Genlock implemented
bAllow	TRUE to allow Genlock, FALSE otherwise
bReturn	TRUE if no error occurred, FALSE otherwise

25.17. SetDetectorGain

HRESULT SetDetectorGain(ULONG dwDetectorGain, ULONG* bReturn)

Function	Set detector gain
dwDetectorGain	Wanted gain
bReturn	TRUE if no error occured, FALSE otherwise

25.18. GetDetectorGain

HRESULT GetDetectorGain(ULONG* pdwDetectorGain, ULONG* bReturn)

Function	Get detector gain
pdwDetectorGain	Current gain
bReturn	TRUE if no error occured, FALSE otherwise

25.19. IsNot4Voice

HRESULT IsNot4Voice(ULONG* pbNot4Voice, ULONG* bReturn)

Function	Is not a four voice detector
pbNot4Voice	TRUE if camera's detector is not a 4 voice, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

25.20. GetDetectorName

HRESULT GetDetectorName(BSTR* pstrName, ULONG* bReturn)

Function	Get detector name
pstrName	Basic string (BSTR) Name
bReturn	TRUE if no error occured, FALSE otherwise

25.21. GetStartup

HRESULT GetStartup(ULONG* pbStartup, ULONG* bReturn)

Function	Get start up option state
pbStartup	TRUE if start up option is activated, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

25.22. SetStartup

HRESULT SetStartup(ULONG bStartup, ULONG* bReturn)

Function	Set start up option
bStartup	When TRUE the camera wait to reach a detector temperature before continuing with live
bReturn	TRUE if no error occured, FALSE otherwise

25.23. IsBolide

HRESULT IsBolide(ULONG* pbBolide, ULONG* bReturn)

Function	Is bolide detector
pbBolide	TRUE if detector is a bolide one
bReturn	TRUE if no error occured, FALSE otherwise

25.24. SetCalibration

HRESULT SetCalibration(ULONG dwDL1, ULONG dwDL2, ULONG dwV1, ULONG dwV2, ULONG dwTemperature, ULONG* bReturn)

Function	Change the calibration parameters
Parameter 1	Digital level 1
Parameter 2	Digital level 2
Parameter 3	Voltage 1
Parameter 4	Voltage 2
Parameter 5	Temperature
Parameter 6	TRUE if no error occured, FALSE otherwise

25.25. GetCalibration

HRESULT GetCalibration(ULONG* pdwDL1, ULONG* pdwDL2, ULONG* pdwV1, ULONG* pdwV2, ULONG* pdwTemperature, ULONG* bReturn)

Function	Give the calibration parameters
Parameter 1	Digital level 1
Parameter 2	Digital level 2
Parameter 3	Voltage 1
Parameter 4	Voltage 2
Parameter 5	Temperature
Parameter 6	TRUE if no error occured, FALSE otherwise

25.26. SetClockContinuous

HRESULT `SetClockContinuous` (ULONG `bContinuous`, ULONG*pbReturn)

Function	Change clock mode
Parameter 1	TRUE for continuous clock, FALSE for mixed clock
Parameter 2	TRUE if no error occured, FALSE otherwise

25.27. GetClockContinuous

HRESULT `GetClockContinuous` (ULONG* `pbContinuous`, ULONG*pbReturn)

Function	Get clock mode
Parameter 1	TRUE if continuous clock, FALSE if mixed clock
Parameter 2	TRUE if no error occured, FALSE otherwise

26. Firmware functions

This section describes functions related to camera firmware.

26.1. GetSoftwareVersion

HRESULT GetSoftwareVersion(ULONG* pdwMajor, ULONG* pdwMinor, ULONG* pdwMinor2, BSTR* pstrDate, ULONG* bReturn)

Function	Get embedded application software version
pdwMajor	Current major version
pdwMinor	Current minor version
pdwMinor2	Current build version
pstrDate	Embedded code date
bReturn	TRUE if no error occured, FALSE otherwise

26.2. OpenSoftwareDlg

HRESULT OpenSoftwareDlg(ULONG* bReturn)

Function	Open firmware upload window
bReturn	TRUE if no error occured, FALSE otherwise

26.3. GoToBoot

HRESULT GoToBoot(ULONG bExecute, ULONG* bReturn)

Function	Go to boot loader mode
bExecute	When TRUE it will automatically execute camera application code in boot loader mode
bReturn	TRUE if no error occured, FALSE otherwise

26.4. GetCodeType

HRESULT GetCodeType(ULONG* pdwCode, ULONG* pdwPlateform, ULONG* bReturn)

Function	Get camera embarked code parameters
pdwCode	Mode : A « Application » B « Boot Loader »
pdwPlateform	Plateform : 0 « Cirrus » 1 « Cassiopea » 2 « Pegasus » 3 « Silver »
bReturn	TRUE if no error occured, FALSE otherwise

26.5. IsApplicationAllowedToUploadCode

HRESULT IsApplicationAllowedToUploadCode(ULONG* pbAllowed, ULONG* bReturn)

Function	Show the capacity to load the code of the application
Parameter 1	Capacity (TRUE/FALSE)
Parameter 2	TRUE if no error occured, FALSE otherwise

26.6. GetSerialNumber

HRESULT GetSerialNumber(BSTR* szSerial, ULONG* bReturn)

Function	Get camera serial number
szSerial	BSTR (Basic string) serial number
bReturn	TRUE if no error occured, FALSE otherwise

26.7. GetBaudRate

HRESULT GetBaudRate(ULONG* pdwUsedBRate, ULONG* bReturn)

Function	Get baud rate
pdwUsedBRate	Current baud rate
bReturn	TRUE if no error occured, FALSE otherwise

26.8. IsInterfaceVersionOk

HRESULT IsInterfaceVersionOk(BSTR* pstrReason, ULONG* bReturn)

Function	Show if the version of the communication interface works
pstrReason	BSTR (Basic string) show the reason why it doesn't work
bReturn	TRUE if no error occured, FALSE otherwise

26.9. IsVideoVersionOk

HRESULT IsVideoVersionOk(BSTR* pstrReason, ULONG* bReturn)

Function	Show if the video translation works
pstrReason	BSTR (Basic string) shows the cause of the dysfunction
bReturn	TRUE if no error occured, FALSE otherwise

27. Backup functions

This section describes functions related to camera setup backup.

27.1. GetLabel

HRESULT GetLabel(LONG nLabel, BSTR* pstrName, ULONG* bReturn)

Function	Give the « label »
nLabel	Number of the current label
pstrName	(BSTR) « Basic string » current label
bReturn	TRUE if no error occured, FALSE otherwise

27.2. SetLabel

HRESULT SetLabel(LONG nLabel, BSTR szName, ULONG* bReturn)

Function	Change le « label »
nLabel	Number of the label wanted
pstrName	(BSTR) « Basic string » label wanted
bReturn	TRUE if no error occured, FALSE otherwise

27.3. Backup

HRESULT Backup(ULONG* bReturn)

Function	Save the current Setup to the static memory of the camera
bReturn	TRUE if no error occured, FALSE otherwise

27.4. SetBackup

HRESULT SetBackup(ULONG bBackUp, ULONG* bReturn)

Function	Change the « Backup » parameters
bBackUp	TRUE to indicate that backup is needed
bReturn	TRUE if no error occured, FALSE otherwise

27.5. GetBackup

HRESULT GetBackup(ULONG* pbBackUp, ULONG* bReturn)

Function	Get back the « Backup » parameters
pbBackUp	TRUE if backup is needed
bReturn	TRUE if no error occured, FALSE otherwise

27.6. GetElapsedTime

HRESULT GetElapsedTime(FLOAT* pfHour, ULONG* pdwCycle, ULONG* bReturn)

Function	Get elapsed time for sterling
pfHour	Current elapsed time
pdwCycle	Current cycle
bReturn	TRUE if no error occured, FALSE otherwise

28. Camera calibration functions

This section describes functions related to Camera Hyper Calibrations Data.

28.1. GetCompCalibParam

HRESULT GetCompCalibParam (ULONG dwTable, ULONG* pdwCalibrationTypeId, FLOAT* pfA, FLOAT* pfB, FLOAT* pfC, FLOAT* pfD, FLOAT* pfE, FLOAT* pfF,ULONG* pbReturn)

Function	Get Hyper calibration parameters
dwTable	Wanted calibration table
pdwCalibrationTypeId	Calibration type 3 : Hyper calibration V2 2 : Hyper calibration 1 : No hyper calibration informations
pfA	…
pfB	…
pfC	…
pfD	…
pfE	…
pfF	
pbReturn	TRUE if no error occured, FALSE otherwise

28.2. GetCompCalibStreamP2

HRESULT GetCompCalibStreamP2(ULONG dwConf, FLOAT* pfStreamP2, ULONG* pbReturn)

Function	Get Hyper calibration V2 parameters
dwConf	Wanted calibration table
pfStreamP2	…
pbReturn	TRUE if no error occured, FALSE otherwise

28.3. GetCompCalibRange

HRESULT GetCompCalibRange (ULONG dwTable, ULONG* pdwCalibrationTypeId, ULONG* pdwValidityDlMax, ULONG* pdwValidityDlMin, FLOAT* pfCalibTempMax, FLOAT* pfCalibTempMin, ULONG* pbReturn)

Function	Get Hyper calibration range
dwTable	Wanted calibration table
pdwCalibrationTypeId	Calibration type 2 : Hyper calibration 1 : No hyper calibration informations
pdwValidityDlMax	Calibration valid down to this value in DL
pdwValidityDlMin	Calibration valid up to this value in DL
pfCalibTempMax	Max calibration in °C
pfCalibTempMin	Min calibration in °C
pbReturn	TRUE if no error occured, FALSE otherwise

28.4. SetCompCalibUpdateParam

HRESULT SetCompCalibUpdateParam(ULONG bUpdate, ULONG* pbReturn)

Function	Set Hyper calibration Update State
bUpdate	TRUE for ON FALSE for OFF
pbReturn	TRUE if no error occured, FALSE otherwise

28.5. GetCompCalibUpdateParam

HRESULT GetCompCalibUpdateParam(ULONG* pbReturn)

Function	Get Hyper calibration Update State
bUpdate	TRUE if update is active FALSE otherwise

29. Camera File Management

This section describes functions related to Camera File management.

29.1. CameraFileOpen

HRESULT CameraFileOpen(ULONG dwOpenFlags, BSTR strFileName, ULONG* pdwHandleFile, ULONG* pdwFileErrorCode, ULONG* pbReturn)

Function	Open Camera File
dwOpenFlags	Open flag bit0 = file mode create bit1 = file mode write bit2 = file mode read
strFileName	File Path
pdwHandleFile	Handle of the opened file
pdwFileErrorCode	File Error Code 0 no error 1 failed 2 unknown disk 3 unknown file 4 invalid name 5 create error 6 invalid file handle 7 invalid parameter 8 invalid mode 9 invalid size 10 invalid cluster 11 disk full 12 access denied 13 write error 14 read error 15 invalid offset 16 end of file
pbReturn	TRUE if no error occured, FALSE otherwise

29.2. CameraFileClose

HRESULT CameraFileClose(ULONG dwHandleFile, ULONG* dwFileErrorCode, ULONG* pbReturn)

Function	Close Camera File
dwHandleFile	Handle of the opened file
dwFileErrorCode	File Error Code 0 no error 1 failed 2 unknown disk 3 unknown file 4 invalid name 5 create error 6 invalid file handle 7 invalid parameter 8 invalid mode 9 invalid size 10 invalid cluster 11 disk full 12 access denied 13 write error 14 read error 15 invalid offset 16 end of file

pbReturn	TRUE if no error occured, FALSE otherwise

29.3. CameraFileRead

HRESULT CameraFileRead(ULONG dwHandleFile, ULONG dwSize, ULONG* dwRealSizeRead, VARIANT FAR* pVarBuffer, ULONG* dwFileErrorCode, ULONG* pbReturn)

Function	Read Camera File
dwHandleFile	Handle of the opened file
dwSize	Size to read
dwRealSizeRead	Final read size
pVarBuffer	Output buffer
dwFileErrorCode	File Error Code 0 no error 1 failed 2 unknown disk 3 unknown file 4 invalid name 5 create error 6 invalid file handle 7 invalid parameter 8 invalid mode 9 invalid size 10 invalid cluster 11 disk full 12 access denied 13 write error 14 read error 15 invalid offset 16 end of file
pbReturn	TRUE if no error occured, FALSE otherwise

29.4. CameraFileWrite

HRESULT CameraFileWrite(ULONG dwHandleFile, ULONG dwSize, VARIANT FAR * pVarBuffer, ULONG* pdwFileErrorCode, ULONG* pbReturn)

Function	Write data to Camera File
dwHandleFile	Handle of the opened file
dwSize	Size to Write
pVarBuffer	Intput Data buffer
dwFileErrorCode	File Error Code 0 no error 1 failed 2 unknown disk 3 unknown file 4 invalid name 5 create error 6 invalid file handle 7 invalid parameter 8 invalid mode 9 invalid size 10 invalid cluster 11 disk full 12 access denied 13 write error 14 read error 15 invalid offset 16 end of file
pbReturn	TRUE if no error occured, FALSE otherwise

29.5. CameraFileSeek

HRESULT CameraFileSeek(ULONG dwHandleFile, LONG nOffset, ULONG dwFrom, ULONG* dwFileErrorCode, ULONG* pbReturn)

Function	Seek Camera File
dwHandleFile	Handle of the opened file
nOffset	Offset (bytes) (offset must a multiple of 4 bytes)
dwFrom	Origin 1 = file begin 2 = file current 3 = file end
dwFileErrorCode	File Error Code 0 no error 1 failed 2 unknown disk 3 unknown file 4 invalid name 5 create error 6 invalid file handle 7 invalid parameter 8 invalid mode 9 invalid size 10 invalid cluster 11 disk full 12 access denied 13 write error 14 read error 15 invalid offset 16 end of file
pbReturn	TRUE if no error occured, FALSE otherwise

29.6. CameraFileEnumerateFirst

HRESULT CameraFileEnumerateFirst(BSTR strFolderName, BSTR* strEnumDiskName, ULONG* dwLinkFolderLevel, ULONG* dwIndexFile, ULONG* dwFileErrorCode, ULONG* pbReturn)

Function	Initialize file enumeration
strEnumDiskName	Folder Name
dwLinkFolderLevel	Id of the link to the folder(To be use with CameraFileEnumerateNext)
dwIndexFile	Output Index of Enumerate (To be use with CameraFileEnumerateNext)
dwFileErrorCode	File Error Code 0 no error 1 failed 2 unknown disk 3 unknown file 4 invalid name 5 create error 6 invalid file handle 7 invalid parameter 8 invalid mode 9 invalid size 10 invalid cluster 11 disk full 12 access denied 13 write error 14 read error 15 invalid offset 16 end of file
pbReturn	TRUE if no error occured, FALSE otherwise

29.7. CameraFileEnumerateNext

HRESULT CameraFileEnumerateNext(BSTR strPreEnumDiskName, ULONG dwPreLinkFolderLevel, ULONG dwPreIndexFile, BSTR* strFileName, ULONG* dwFileStatusMask, ULONG* dwFileSize, ULONG* dwIndexFile, ULONG* dwFileErrorCode, ULONG* pbReturn)

Function	File enumeration
strPreEnumDiskName	Folder Name (returned by CameraFileEnumerateFirst)
dwPreLinkFolderLevel	Id of the link to the folder(returned by CameraFileEnumerateFirst)
dwIndexFile	Index of Enumerate (returned by CameraFileEnumerateFirst or CameraFileEnumerateNext)
strFileName	File Name
dwFileStatusMask	File status mask bit0 = valid name bit1 = file bit2 = folder
dwIndexFile	Output Index of Enumerate (To be use with CameraFileEnumerateNext)
dwFileErrorCode	File Error Code 0 no error 1 failed 2 unknown disk 3 unknown file 4 invalid name 5 create error 6 invalid file handle 7 invalid parameter 8 invalid mode 9 invalid size 10 invalid cluster 11 disk full 12 access denied 13 write error 14 read error 15 invalid offset 16 end of file
pbReturn	TRUE if no error occured, FALSE otherwise

29.8. CameraFileDelete

HRESULT CameraFileDelete(BSTR strFileName, ULONG* dwFileErrorCode, ULONG* pbReturn)

Function	Delete File
strFileName	Full Path of the NUC to delete
dwFileErrorCode	File Error Code 0 no error 1 failed 2 unknown disk 3 unknown file 4 invalid name 5 create error 6 invalid file handle 7 invalid parameter 8 invalid mode 9 invalid size 10 invalid cluster 11 disk full 12 access denied 13 write error 14 read error 15 invalid offset 16 end of file
pbReturn	TRUE if no error occured, FALSE otherwise

29.9. CameraFileStatus

HRESULT CameraFileStatus(ULONG dwHandleFile, ULONG* dwFileSize, ULONG* dwCurPosition, ULONG* dwFileErrorCode, ULONG* pbReturn)

Function	Get File Status
dwHandleFile	Handle of the target file
dwFileSize	Current File Size (bytes)
dwCurPosition	Current Position (bytes)
dwFileErrorCode	File Error Code 0 no error 1 failed 2 unknown disk 3 unknown file 4 invalid name 5 create error 6 invalid file handle 7 invalid parameter 8 invalid mode 9 invalid size 10 invalid cluster 11 disk full 12 access denied 13 write error 14 read error 15 invalid offset 16 end of file
pbReturn	TRUE if no error occured, FALSE otherwise

29.10. CameraFileDiskStatus

HRESULT CameraFileDiskStatus(BSTR strDiskName, ULONG* dwDiskTypeId, ULONG* bIsDataValid, ULONG* dwOpenedFile, ULONG* dwFreeSpace, ULONG* dwFreeNameFile, ULONG* dwFileErrorCode, ULONG* pbReturn)

Function	Get Disk Status
strDiskName	Internal Camera Disk name (FlashDisk)
dwDiskTypeId	Return Disk type 1 = Flash
bIsDataValid	TRUE if Valid FALSE otherwise
dwOpenedFile	Current Number of Opened file
dwFreeSpace	Current Disk free space
dwFreeNameFile	Current Number of free Name File
dwFileErrorCode	File Error Code 0 no error 1 failed 2 unknown disk 3 unknown file 4 invalid name 5 create error 6 invalid file handle 7 invalid parameter 8 invalid mode 9 invalid size 10 invalid cluster 11 disk full 12 access denied 13 write error 14 read error 15 invalid offset 16 end of file
pbReturn	TRUE if no error occurred, FALSE otherwise

29.11. CameraFileRename

HRESULT CameraFileRename(BSTR strCurrentFileName, BSTR strNewFileName, ULONG* dwFileErrorCode, ULONG* pbReturn)

Function	Rename Camera file
strCurrentFileName	Original file name
strNewFileName	New file name
dwFileErrorCode	File Error Code 0 no error 1 failed 2 unknown disk 3 unknown file 4 invalid name 5 create error 6 invalid file handle 7 invalid parameter 8 invalid mode 9 invalid size 10 invalid cluster 11 disk full 12 access denied 13 write error 14 read error 15 invalid offset 16 end of file
pbReturn	TRUE if no error occurred, FALSE otherwise

29.12. CameraFileFormat

HRESULT CameraFileFormat(BSTR strDiskName, ULONG bFormat, ULONG* dwFileErrorCode, ULONG* pbReturn)

Function	Format Camera Disk **warning, all data are deleted!**
strDiskName	Internal Camera Disk name (FlashDisk)
bFormat	TRUE to format FALSE otherwise
dwFileErrorCode	File Error Code 0 no error 1 failed 2 unknown disk 3 unknown file 4 invalid name 5 create error 6 invalid file handle 7 invalid parameter 8 invalid mode 9 invalid size 10 invalid cluster 11 disk full 12 access denied 13 write error 14 read error 15 invalid offset 16 end of file
pbReturn	TRUE if no error occurred, FALSE otherwise

29.13. CameraFileDownloadFromCamera

HRESULT CameraFileDownloadFromCamera(BSTR strCameraFile, BSTR strRealPcFile, ULONG* dwRealFileSize, BSTR* strFileMapping, ULONG* dwHandleInternalFileMapping, ULONG* dwFileErrorCode, ULONG* pbReturn)

Function	Download File From Camera
strCameraFile	Camera File Name
strRealPcFile	Computer destination file
dwRealFileSize	File Size
strFileMapping	Internal file Mapping title (use to access Mapping memory)
dwHandleInternalFileMapping	Internal Handle of file Mapping (use to close file mapping)
dwFileErrorCode	File Error Code 0 no error 1 failed 2 unknown disk 3 unknown file 4 invalid name 5 create error 6 invalid file handle 7 invalid parameter 8 invalid mode 9 invalid size 10 invalid cluster 11 disk full 12 access denied 13 write error 14 read error 15 invalid offset 16 end of file
pbReturn	TRUE if no error occurred, FALSE otherwise

29.14. CameraFileUploadToCamera

HRESULT CameraFileUploadToCamera(BSTR strCameraFile, BSTR strRealPcFile, ULONG dwRealFileSize, BSTR strFileMapping, ULONG* dwFileErrorCode, ULONG* pbReturn)

Function	Download File From Camera
strCameraFile	Camera File Name
strRealPcFile	Computer source file
dwRealFileSize	File Size
strFileMapping	Source file Mapping title (use to access Mapping memory)
dwFileErrorCode	File Error Code 0 no error 1 failed 2 unknown disk 3 unknown file 4 invalid name 5 create error 6 invalid file handle 7 invalid parameter 8 invalid mode 9 invalid size 10 invalid cluster 11 disk full 12 access denied 13 write error 14 read error 15 invalid offset 16 end of file
pbReturn	TRUE if no error occurred, FALSE otherwise

29.15. CameraFileCloseInternalFileMapping

HRESULT CameraFileCloseInternalFileMapping(ULONG dwHandleInternalFileMapping, ULONG* pbReturn)

Function	Close Internal File Mapping
dwHandleInternalFileMapping	Internal Handle of file Mapping (returned by CameraFileDownloadFromCamera)
pbReturn	TRUE if no error occurred, FALSE otherwise

30. Virtual Camera window message

Virtual camera sends CEDIP message to inform that some parameters has changed. Include the header MessageDotNet.h and register the cedip window message (CM_BROADCAST_NAME). Then use WPARAM of cedip message to identify an event.

Example :

```
m_wMsgCedip          = RegisterWindowMessage(CM_BROADCAST_NAME);

WindowProc(UINT message, WPARAM wParam, LPARAM lParam)
{

      if (message == m_wMsgCEDIP)
      {
               switch(wParam)
               {
                        case CM_CONFIG_CHANGED                : DoAction();break;
                                 ...
                                 ...
                                 ...
                                 ...
                        case CM_NBR_PACKET_CHANGED  : DoAction2();break;

                        default:break;
               }
      }
}
```

WPARAM	Description
CM_CONFIG_CHANGED	One the camera parameter has changed
CM_CONFIG_SAVED	NUC configuration has been saved in flash memory.
CM_CONFIG_NAME_CHANGED	NUC configuration label changed
CM_CONFIG_AUTO_SYNCHRONIZE_CHANGED	Synchronization with Nuc changed
CM_WINDOW_CHANGING	Windowing would changed
CM_WINDOW_CHANGED	Windowing changed
CM_WINDOW_CURRENT_CHANGED	Current windowing changed
CM_FRAME_RATE_CHANGED	Frame rate changed
CM_FRAME_RATE_MAX_CHANGED	Frame rate max changed
CM_EXTERNAL_CHANGED	External trigger state changed
CM_GENLOCK_CHANGED	GenLock state changed
CM_GENLOCK_ALLOWED_CHANGED	Gen Lock Capability changed
CM_TTL_ALLOWED_CHANGED	TTL Capability changed
CM_CAN_CHANGE_FRAME_RATE_CHANGED	Set frame rate capability changed
CM_TRIGGER_OUT_CHANGED	Detector trigger out changed
CM_CURRENT_TI_CHANGED	Current integration time changed
CM_CURRENT_VIDEO_CHANGED	Current video chanel changed
CM_MULTI_TI_CHANGED	Multi IT mode changed (Multi/Mono)
CM_FILTER_NAME_CHANGED	Filter name changed
CM_CURRENT_FILTER_CHANGED	Current wheel position changed
CM_BB_PRESENT_CHANGED	Internal black body presence changed

CM_ORION_MODE_CHANGED	Orion mode changed
CM_ZOOM_CHANGED	Video zoom active changed
CM_RETICLE_CHANGED	Reticle mode changed
CM_VIDEO_ON_OF_CHANGED	Video output state (enable/disable) changed
CM_VIDEO_RESOLUTION_CHANGED	Video resolution changed
CM_VIDEO_LIVE_CHANGED	Video out put live changed
CM_VIDEO_FLIPH_CHANGED	Horizontal flip changed
CM_VIDEO_FLIPV_CHANGED	Vertical flip changed
CM_VIDEO_MAP_CHANGED	Video mapping changed
CM_VIDEO_ROTATION_CHANGED	Video rotation changed
CM_VIDEO_ROI_CHANGED	Video Region Of Interest changed
CM_NUC_CHANGED	Nuc changed
CM_NUC_SHUTTER_CHANGED	Use of shutter for NUC changed
CM_AUTO_GAIN_CHANGED	AGC enable changed
CM_AGC_PERCENTAGE_CHANGED	AGC percentage changed
CM_AGC_HYSTERESIS_CHANGED	AGC hysterisis changed
CM_AGC_LIMIT_CHANGED	AGC limit changed
CM_AGC_SMOOTH_CHANGED	AGC smooth changed
CM_OFFSET_LIMIT_CHANGED	Offset limit changed
CM_GAMMA_CORRECTION_CHANGED	Gamma correction changed
CM_ROI_CHANGED	Region Of Interest changed
CM_PALETTE_CHANGED	Palette changed
CM_OFFSET_GAIN_CHANGED	Gain or offset changed
CM_DEBUG_CHANGED	Video debug state(enable/disable) changed
CM_IS_DISCONNECTED	Communication lost
CM_DONT_RETRY_CONNECTION	Communication aborted
CM_NBR_PACKET_CHANGED	Number of frame by packet changed
CM_DELAY_LOCKIN_CHANGED	Lockin delay changed
CM_LOCKIN_THRESHOLD_CHANGED	Lockin threshold or signal changed
CM_LOCKIN_PARAM_CHANGED	Phase or N factor changed
CM_BAUD_RATE_CHANGED	Baud rate changed
CM_FOCAL_CHANGED	Focal changed
CM_CONTINUOUS_CLOCK_CHANGED	Clock mode changed
CM_LENS_CHANGED	Lens changed
CM_SERIAL_NUMBER_CHANGED	Serial number changed
CM_COMPENSATION_CHANGED	Compensation changed
CM_CNUC_CALIB_CHANGE	Calibration parameters changed

31. Getting started with calibration interface using C++

This section focus only on C++ language.

In order to connect to use calibration tool through Vircam, it is first needed to instantiate the distributed CVirtualCam object.

- Include the following files:

```
#include "_VCamServer_i.c"
#include "_VCamServer.h"
```

- Create an interface pointer:

```
IVcCalibration* m_pVcCalibration;
```

- Create an instance of the CVcCalibration Object:

```
// Initialize COM library
HRESULT hr = CoInitializeEx(NULL, COINIT_MULTITHREADED);
if (FAILED(hr)) //if initialization fails then exit
{
     AfxMessageBox("Failed to initialize COM library");
}

m_pVcCalibration = NULL;

//create the com
hr = CoCreateInstance(     CLSID_CVcCalibration, NULL,
                           CLSCTX_LOCAL_SERVER , IID_IVcCalibration,
                           (void**) & m_pVcCalibration);
```

- Freeing the instance is done as following:

```
if (m_pVcCalibration){
     m_pVcCalibration->Release();
     m_pVcCalibration = NULL;
}
CoUninitialize();
```

For more details see VirtualClientDll sources.

32. Calibration functions

This section describes functions related to calibration.

32.1. OpenFile

HRESULT OpenFile (BSTR strFile, ULONG* bReturn)

Function	Open calibration file
strFile	BSTR (Basic string) Calibration file
bReturn	TRUE if no error occured, FALSE otherwise

32.2. CreateHyperCalibration

HRESULT CreateHyperCalibration (void)

Function	Create Hyper Calibration and replace file calibration data by hyper calibration parameters.

32.3. SetHyperCalibrationParam

HRESULT SetHyperCalibrationParam (ULONG dwCalibrationTypeId, FLOAT fA, FLOAT fB, FLOAT fC, FLOAT fD, FLOAT fE, FLOAT fF, ULONG dwValidityDlMax, ULONG dwValidityDlMin, FLOAT fCalibTempMax, FLOAT fCalibTempMin, ULONG* pbRetrun)

Function	Set hyper calibration parameters.
dwCalibrationTypeId	Calibration type 2 : Hyper calibration 1 : No hyper calibration informations
fA	…
fB	…
fC	…
fD	…
fE	…
fF	…
dwValidityDlMax	Calibration valid down to this value in DL
dwValidityDlMin	Calibration valid up to this value in DL
fCalibTempMax	Max calibration in °C
fCalibTempMin	Min calibration in °C
pbReturn	TRUE if no error occured, FALSE otherwise

32.4. SetIntegration

HRESULT SetIntegration (ULONG dwIntegration, ULONG* pbReturn)

Function	Set integration time for calibration calcul
dwIntegration	Integration time
pbReturn	TRUE if no error occured, FALSE otherwise

32.5. SetCameraTemperature

HRESULT SetCameraTemperature (FLOAT fCamTemp, ULONG* bReturn)

Function	Set Camera temperature which is used to update the calibration depending on the camera temperature.
fCamTemp	Caméra temperature in kelvin
bReturn	TRUE if no error occured, FALSE otherwise

32.6. SetRadiometryParam

HRESULT SetRadiometryParam (FLOAT fEmissivity, FLOAT fTempBackGround, FLOAT fTransmission, FLOAT fTempAtmosphere , ULONG bExtrapolate, ULONG* bReturn)

Function	Set radiometry parameters
fEmissivity	Emissivity of the object
fTempBackGround	Background temperature
fTransmission	Atmospheric transmission
fTempAtmosphere	Atmosphere's temperature
bExtrapolate	TRUE to activate the extrapolation, FALSE otherwise
bReturn	TRUE if no error occured, FALSE otherwise

32.7. ConvertDlToTemp

HRESULT ConvertDlToTemp (ULONG dwPoint, ULONG dwLine, ULONG dwMaxLevel, BSTR strDlBuffer, BSTR strTempBuffer, ULONG* bReturn)

Function	Get temperature from digital level
dwPoint	Number of point of the image
dwLine	Number of line of the image
dwMaxLevel	Maximum Digital Level
strDlBuffer	BSTR (Basic string) name of shared memory containing the image in DL
strTempBuffer	BSTR (Basic string) name of shared memory to full with the image in temperature
bReturn	TRUE if no error occured, FALSE otherwise

32.8. ConvertTempToDl

HRESULT ConvertTempToDl (FLOAT fTemp, USHORT* pwDl, ULONG* pbReturn)

Function	Get digital level from temperature
fTemp	Temperature to convert (in °C)
pwDl	Result in Digital Level
pbReturn	TRUE if no error occured, FALSE otherwise

32.9. GetUnitX

HRESULT GetUnitX (BSTR* pstrUnitX, ULONG* bReturn)

Function	Get X unit name
pstrUnitX	BSTR (Basic string) name of unit x
bReturn	TRUE if no error occured, FALSE otherwise

32.10. GetUnitY

HRESULT GetUnitX (BSTR* pstrUnitY, ULONG* bReturn)

Function	Get Y unit name
pstrUnitY	BSTR (Basic string) name of unit Y
bReturn	TRUE if no error occured, FALSE otherwise