Penguin / Pro SDK Manual
Version 1.2.21

Introduction and Overview

Version 1.2.21 of the Pixera Penguin / Pro Software Developer's Kit (SDK) lets you develop your own Windows WDM (Windows Driver Model) based application to control the Pixera Penguin / Pro Monochrome or Color cameras, capturing images singly and displayed as a motion sequence, for your own custom image capture, viewfinder, and processing requirements.
You should already have the Pixera Penguin / Pro system, including the Pixera (monochrome or color) camera, PCI card interface adapter, and software applications. You should also have your own C++ language Microsoft Windows Studio or Visual C++ development tools.
You can create your application from scratch or modify the fully functional sample application, PixTestApp, provided in this kit. The PixTestApp sample application exercises most of the API calls and demonstrates the implementation of a viewfinder application.
The PixTestApp.exe Windows 98/2000/ME application is a compilation of the PixTestApp.cpp source code included in this kit, which lets you run the sample application immediately.
The Dynamic Link Library (.dll) files provided with the Pixera Penguin / Pro Software Development Kit, PixSDK.dll, provide low-level camera control, image processing, and image retrieval functions.
The accompanying header files provide the application interface to the PixSDK.dll file, which your application should load in its initialization routines.

Pixera Penguin / Pro SDK Version 1.2.21 Inventory

The Pixera Penguin / Pro Software Developer's Kit (SDK) comes with this user guide and a development kit CD.
The development CD (and the downloadable version) includes the following files:

PixSDK.dll Pixera Penguin / Pro library module for Windows 98/2000/ME
PixSDK.h Pixera Penguin / Pro header file
PixTestApp project Sample Windows Visual C++ Version 6.0 Pixera Penguin / Pro application source code.
PixTestApp.exe Windows 98/2000/ME sample Pixera Camera Control application executable
ReadMe.txt A text file identifying the exact version number of this SDK
PixSDKDocs.txt A text file containing the text of this document
PixSDKDocs.doc A Microsoft Word 97 file of this document

Installing the Pixera Penguin / Pro SDK

You should have the Pixera Penguin / Pro System, which provides the Pixera camera, PCI bus interface adapter, and software applications as well as a Microsoft C language or C++ language compiler for Windows 98/2000/ME along with standard development tools such as an editor and debugger.
Installing the Pixera Camera hardware
First install the Pixera Camera hardware on your system, then install the appropriate Pixera device driver from the corresponding operating

system folder on the Penguin / Pro applications CD, under the device drivers folder. .

Installing the PCI bus interface adapter

Turn off your computer. Pixera Corporation technical support recommends that you remove the power cable to be sure power remains off.

Open the computer case so you can see the PCI bus slots. (Most PCI computers also support ISA or EISA slots as well: be sure to match your Pixera Camera PCI bus interface adapter to the PCI slots.)

Install the Pixera Camera PCI bus interface adapter in an available slot. Be sure the PCI bus interface adapter is correctly aligned and tightened down.

Close up the computer case. Use the supplied cable to connect the Pixera Camera to the bus interface adapter.

Never unplug the camera cable from either the camera side or the PCI card side while the PC is powered as this may damage the camera.

You should be ready to power on, boot up Windows 98/2000/ME, and install the corresponding Penguin / Pro camera device driver.

Installing the Pixera Penguin / Pro device driver

When your Pixera Camera hardware is installed, turn your computer’s power on, boot up Windows 98/2000/ME.

Windows 98, 2000 or ME will ask you for the Vendor’s device driver.

Select the appropriate .inf file from the corresponding operating system folder under Device Drivers on the application software CD that comes with the Penguin camera.

Installing the Pixera Penguin / Pro SDK software

Create a Pixera Penguin / Pro development directory on the computer that will host your development work.

Copy the SDK files and folders to the Pixera Penguin / Pro development directory.

When you’ve got the Pixera Penguin / Pro SDK installed, you’re ready to run the PixTestApp.exe program or start developing your own application.

To open the PixTestApp project under Visuall C++ Version 6.0 or later, first make sure that Visual C++ 6.0 is installed on the computer, then open the PixTestApp.dsw file.

Using the Pixera Penguin / Pro SDK

Use your Visual C or C++ development tools to create your Pixera Penguin / Pro application.

Trying the PixTestApp.exe program

If you wish, you can run the PixTestApp.exe application to get an idea of how a rudimentary Pixera Penguin / Pro application works. When you run the PixTestApp.exe program, it will load the dll and WDM files automatically and as you click appropriate buttons. (Be sure you have installed and connected the Pixera Camera hardware before you turn your computer's power on.) In order for the PixSDK dll library to function correctly, you must also provide the 6 data files (with .dat extensions) in the same folder as the executable. Try out the buttons to start up the Pixera Camera, view and capture an image.

There are a row of lettered buttons to the right of the toolbar of the PixTestApp test application. These are currently assigned to function calls testing various focus finder and levels adjustment functions. Please feel free to modify the button handling code in the file PixTestAppView.cpp file in order to customize and test various SDK functionality.

How things work together

Your application will provide the user-interface to the Pixera Camera software and hardware system. Its buttons and menus should correspond to features in the PixSDK.dll module.

Refer to the reference section in this guide as well as the PixSDK.h to write your own functions that call standard C language API functions in the PixSDK.dll module.

The PixSDK.dll module dynamically loads and unloads the Penguin.sys WDM device driver through CreateFile() calls. To view and capture images, the PixSDK.dll module makes calls to the PixSDK.dll pre-capture processing module which formats the memory image returned from the Penguin.sys device driver.

Application Development

The PixTestApp project was developed using the Microsoft Foundation Class (MFC) libraries and methodology. For more information on MFC development please refer to the appropriate Microsoft Visual C++ documentation. You do not need to program in C++ or utilize MFC in order to take advantage of the PixSDK library, but it may be necessary in order to understand the functionality of the PixTestApp application source code.

PixTestAppView.cpp contains functions which respond to user interface events, or make calls into the PixSDK library to prepare for these events:

CPixTestAppView() constructor demonsrates loading the driver and checking if the camera is connected.

~CPixTestAppView() destructor demonstrates stopping the motion preview, disengaging the camera, and unloading the driver.

OnDraw(), in the sample application, and CAM_MIA_StartDraw(), in the SDK, are only included for legacy purposes and are no longer recommended to use
CAM_MIA_StartAutoDraw() will now automatically draw the live preview image into the current window using "Video for Windows."

OnPreview() starts the Viewfinder motion preview, and it illustrates how to call CAM_MIA_StartAutoDraw() by passing a global image DIB handle and the device context for drawing the image into the current window, which is obtained using GetDC()->GetSafeHdc(). When the application window was created in the application's PreCreateWindow() function it included extra window space for a 696 x 520 preview image. Once CAM_MIA_StartAutoDraw() is called it will automatically draw the live preview image into the window at the fastest possible frame rate. No other functions are required.

Setting Viewfinder resolution is demonstrated with the functions: OnViewfinderZoom(), OnViewfinderFast(), and OnViewfinderFull()

Capturing a still image: OnCapture() and OnCaptureDone()

Capturing a viewfinder preview image from the motion sequence image stream: OnPreviewCapture() and OnPreviewCaptureDone().

What's New in Versions 1.2.18 through 1.2.21

Fixed problem of CAM_BB_SetFactors() interacting with auto white balance mode.

Deactivate color transform from default behavior for Brightfield mode

Fix monochrome still capture and motion image brightness problems.

What's New in Versions 1.2.12 through 1.2.17

Fixed problem of shifted image in captured 8-bit monochrome DIB image.

Fixed problem of single bright image which occured when starting MIA

Fixed problem where first image captured after MIA was black.

Manual exposure mode is now set by default after calling CAM_SIA_PrepXXX().

Maximum framerate is now possible when calling the sequence CAM_SIA_CaptureExpose() and CAM_SIA_CaptureProcess() repeatedly.

MIA mode may be activated between calls to CAM_SIA_PrepXXXX() and CAM_SIA_CaptureExpose() without side effects.

What's New in Versions 1.2.9 through 1.2.11

Added functions:

CAM_SIA_Prep8Bit(HANDLE)
CAM_SIA_Prep16Bit(HANDLE)
CAM_SIA_Prep24Bit(HANDLE)
CAM_SIA_Prep48Bit(HANDLE)
CAM_SIA_CaptureExpose()
CAM_SIA_CaptureProcess()

This allows image capturing to be setup prior to the actual expose phase of the capture, allowing the light source to be turned on and off immediately before and after the call to CAM_SIA_CaptureExpose(). CAM_SIA_CaptureExpose() and CAM_SIA_CaptureProcess() sequencess may be called any number of times after CAM_SIA_PrepXXXXX() in order to minimize processing and maximize the framerate.

CAM_BB_SetFactors() now functions correctly, even when preview mode is not active.

Bug fixed in processing of monochrome camera 1392 x 1040 resolution capture allowing sharpest image possible.

Captured image now incorporates the white balance settings, even if the white balance factors were not set in preview mode.

What's New in Version 1.2.8

Using drawing functions other than those in Video for Windows library to eliminate crashing problems in CAM_MIA_StartAutoDraw().

Include a new SDK function, CAM_FF_GetFineValue(), which returns a focus finder value between 0 and 1000, providing more sensitivity than the function CAM_FF_GetEvalValue().

What's New in Versions 1.2.2 through 1.2.7

Handles are locked down to reduce access violation errors, particularly in memory intensive applications.

Still captures can now be performed even when MIA (motion preview) is not active.

Problem is fixed where calling CAM_MIA_Start() after CAM_MIA_StartAutoDraw() and CAM_MIA_Stop() prevented the CAM_MIA callback functions from working properly.

Fixed a problem where CAM_MIA_Stop() accessed a heap block which had been previously freed.

Fixed CAM_BB_GetFactors() and CAM_BB_SetFactors() so that they work properly.

Fixed a problem so that CAM_SIA_GetResolution() returns proper values for VGA mode.

What's New in Version 1.2.1

At the time CAM_Init() is called, the current working directory is saved and used when accessing the .dat files from that point on.

What's New in Version 1.2.0

CAM_Init(colorMode) and CAM_IP_SetColor(colorMode) SDK functions now take a single argument which specifies one of three color modes depending on the camera being used, or whether color images are desired:

kMonochromeCam - specifies that monochrome images are to be captured or displayed from the monochrome camera.

kColorCamMono - specifies that monochrome images are to be captured or displayed from the color camera.

kColorCamColor - specifies that color images are to be captured or displayed from the color camera.

There are also two new capture functions which are only useful with the monochrome camera:

CAM_SIA_Start8Bit() and CAM_SIA_Start16Bit() will capture 8-bit and 16-bit bitmaps respectively, but only in kMonochromeCam mode.

When an 8-bit bitmap is passed to CAM_IMG_Save(), as in the the test application, then it will be saved as an 8-bit monochrome BMP or TIFF image, whichever is specified. When a 16-bit bitmap is passed to CAM_IMG_Save() then it can be saved as a 16-bit TIFF image.

Pixera Penguin / Pro SDK Reference

The following functions comprise the useful C language API calls an application should use to control the Pixera Penguin / Pro Camera and capture images. Function calls are presented in related groups.

Load and set up driver

CAM_LoadDriver
CAM_UnloadDriver
CAM_IsDriverLoaded

Get Software Versions

CAM_GetVersionOfDriver
CAM_GetVersion

Set up camera condition

CAM_Init
CAM_Exit
CAM_IsConnected

Automatic exposure and photometry

CAM_AE_SetSpotSize

CAM_AE_GetSpotSize
CAM_AE_SetSpotPosition
CAM_AE_GetSpotPosition
CAM_AE_Start
CAM_AE_Stop
CAM_AE_Lock
CAM_AE_GetMode
CAM_AE_GetStatus
CAM_AE_SetAdjust
CAM_AE_GetAdjust
CAM_AE_SetSpotColor
CAM_AE_GetSpotColor
CAM_AE_SetMicroMode
CAM_AE_GetMicroMode

Color Balance

CAM_WB_SetMode
CAM_WB_GetMode
CAM_WB_SetRegion
CAM_WB_GetRegion
CAM_WB_SetFactors
CAM_WB_GetFactors
CAM_WB_CalibrateFactors
CAM_BB_Start
CAM_BB_Stop
CAM_BB_SetRegion
CAM_BB_GetRegion
CAM_BB_GetFactors
CAM_BB_CalibrateFactors

Exposure Condition

CAM_EXP_SetSpeed
CAM_EXP_GetSpeed
CAM_EXP_SetSensitivity
CAM_EXP_GetSensitivity

Image Processing

CAM_IP_SetLuminanceLevels
CAM_IP_GetLuminanceLevels
CAM_IP_SetOrientation
CAM_IP_GetOrientation
CAM_IP_ColorCapture
CAM_IP_IsColorCapture

Focus Finder

CAM_FF_Start
CAM_FF_Stop
CAM_FF_IsRunning
CAM_FF_GetEvalValue
CAM_FF_GetFineValue
CAM_FF_Reset
CAM_FF_SetRegion
CAM_FF_GetRegion

Motion Observation and Acquisition

CAM_MIA_SetResolution
CAM_MIA_GetResolution
CAM_MIA_GetPixelSize
CAM_MIA_StartAutoDraw
CAM_MIA_Start [obsolete]
CAM_MIA_Stop
CAM_MIA_IsRunning
CAM_MIA_Capture
CAM_MIA_SetHDC

Capture and save a Still Image
CAM_SIA_SetResolution
CAM_SIA_GetResolution
CAM_SIA_GetPixelSize
CAM_SIA_SetAccumulateTimes
CAM_SIA_GetAccumulateTimes
CAM_SIA_SetAccumulateMode
CAM_SIA_GetAccumulateMode
CAM_SIA_Start8Bit
CAM_SIA_Start16Bit
CAM_SIA_Start24Bit
CAM_SIA_Start48Bit
CAM_SIA_IsRunning
CAM_IMG_Save

CAM_LoadDriver — **Load and set up driver**

BOOL CAM_LoadDriver (void)

Under Windows 98/2000/ME, dynamically loads the WDM driver.

Parameters
None

Return Values
TRUE if successful
FALSE otherwise

CAM_UnloadDriver — **Unload device driver**

vod CAM_UnloadDriver (void)

Under Windows 98/2000/ME, dynamically unloads the WDM driver.

Parameters
None

CAM_IsDriverLoaded — **Check if driver loaded**

BOOL CAM_IsDriverLoaded (void)

Checks whether the device driver has been already loaded.

Parameters
None

Return Values
TRUE Device driver has been already loaded.
FALSE Device driver is not loaded yet.

CAM_GetVersionOfDriver — **Get current driver version**

void CAM_GetVersionOfDriver(short* pMajorVersionOut, short* pMinorVersion, short * pBugfixVersionOut)

Returns the major, minor and bug fix version numbers of the camera driver

Parameters

OUT   pMajorVersionOut - Major version release
OUT   pMinorVersionOut - Minor version upgrade
OUT   pBugfixVersionOut - Bug fix version upgrade.

Return Values
      None

CAM_GetVersion **Get current SDK version**

void CAM_GetVersionOfDriver(short* pMajorVersionOut, short* pMinorVersionOut, short * pBugfixVersionOut)

Returns the major, minor and bug fix version numbers of the SDK DLL

Parameters
OUT   pMajorVersionOut - Major version release
OUT   pMinorVersionOut - Minor version upgrade
OUT   pBugfixVersionOut - Bug fix version upgrade.

Return Values
      None

CAM_Init — Initialize camera

```
void CAM_Init (ColorModeEx    theColorMode)
```

Initializes the camera.

Parameters

IN theColorMode - specifies the color mode and type of camera used

It can be any one of the following values.

Value	Meaning
kMonochromeCam	Monochrome camera
kColorCamMono	Color camera in monochrome mode
kColorCamColor	Color camera in color mode

Return Values

None

CAM_Exit — Disengage camera

```
void CAM_Exit (void)
```

Ends use of the camera.

Parameters

None

Return Values

None.

CAM_IsConnected — Check if camera connected

```
BOOL CAM_IsConnected (void)
```

Checks the connection between the camera and the cable.

Parameters

None

Return Values

TRUE	Camera is connected to the camera cable.
FALSE	Camera is disconnected to the camera cable.

CAM_AE_SetSpotSize — Set AE spot params

```
void CAM_AE_SetSpotSize(bool bDrawingOnOff, PxRectEx *rc,
UCHAR ucRed, UCHAR ucBlue, UCHAR ucGreen);
```

Sets the photometry region.

Parameters

IN bDrawingOnOff - Specifies the flag which specifies if SDK draws the photometry region.

IN rc - Specifies the photometry region.

IN ucRed - Specifies the Red value of the frame color for the photometry region.

IN ucBlue - Specifies the Blue value of the frame color for the photometry region

IN ucGreen - Specifies the Green value of the frame color for the photometry region

Return Values
None

CAM_AE_GetSpotSize — Get AE spot params

void CAM_AE_GetSpotSize(bool *pbDrawingOnOff, PxRectEx *pRC, UCHAR *pRed, UCHAR *pGreen, UCHAR *pBlue);

Gets the area of photometry.

Parameters

OUT pbDrawingOnOff - Specify a buffer retrieving the flag which specifies if the photometry region is drawn.
OUT pRC - Specify a buffer retrieving the photometry region.

OUT pRed - Specify a buffer retrieving Red value of the frame color of photometry region.

OUT pGreen - Specify a buffer retrieving Green value of the frame color of photometry region.

OUT pBlue - Specify a buffer retrieving Red value of the frame color of photometry region..

Return Values
None.

Remarks
This function retrieves the conditions set by CAM_AE_SetSpotSize () function.

CAM_AE_SetSpotPosition — Set AE spot position

void CAM_AE_SetSpotPosition (bool bDrawingOnOff, PxRectEx *rc, UCHAR ucRed, UCHAR ucBlue, UCHAR ucGreen);

Sazme as CAM_AE_SetSpotSize()

CAM_AE_GetSpotPosition — Get AE spot position

void CAM_AE_GetSpotPosition (bool *pbDrawingOnOff, PxRectEx *pRC, UCHAR *pRed, UCHAR *pGreen, UCHAR *pBlue);

Same as CAM_AE_GetSpotSize()

CAM_AE_Start — Start AE processing

void CAM_AE_Start (void)

Starts the auto exposure control.

Parameters
None

Return Values

None.

CAM_AE_Stop **Stop AE processing**

void CAM_AE_Stop (void)

Stops the automatic exposure control.

Parameters
None

Return Values

None.

Remarks
When this function calls, stops the automatic exposure control and sets the last computed exposure time as manual exposure time.

CAM_AE_Lock **Lock current AE setting**

void CAM_AE_Lock (bool bLock)

Locks or unlocks the automatic exposure control.

Parameters
IN bLock Specifies the lock control of the automatic exposure. It can be any one of the following values.

Value	Meaning
TRUE	Locks the auto exposure control by current conditions.
FALSE	Releases the AE-lock, and restarts automatic exposure control.

Return Values
None

CAM_AE_GetMode **Get current AE mode**

AutoExposureModeEx CAM_AE_GetMode(void)

Gets the mode of automatic exposure control.

Parameters
None

Return Values
The return value is a mode of automatic exposure control.
It can be any one of the following values.

Value	Meaning
kAEUnlocked	Automatic exposure control is active.
kAELocked	Locked the automatic exposure control.

kManualExposure Automatic exposure control is stopped. (manual exposure control)

CAM_AE_GetStatus — Get current AE status

AutoExposureStatusEx CAM_AE_GetStatus();

Gets status of the automatic exposure control.

Parameters
None

Return Values
The return value is a status of the automatic exposure control.
It can be any one of the following values.

Value	Meaning
kAENotFunctional	Automatic exposure control doesn't work.
KAEUnderExposed	Computed exposure time is shorter than appropriate exposure time. (underexposure)
KAEGoodExposure	Computed exposure time is appropriate.
KAEOverExposed	Computed exposure time is longer than appropriate exposure time. (overexposure)

CAM_AE_SetAdjust — Set AE compensation

BOOL CAM_AE_SetAdjust (int nCoefficient)

Sets the compensation coefficient of automatic exposure control.
nCoefficient parameter means 2nCoefficient / 3.

Parameters
IN nCoefficient Specifies the compensation coefficient of automatic exposure control.
It takes from -6 to +6.

Return Values

TRUE if successful
FALSE otherwise.

CAM_AE_GetAdjust — Get AE adjust compensation

int CAM_AE_GetAdjust (void)

Gets the compensation coefficient of automatic exposure control.

Parameters
None

Return Values
The return value is the compensation coefficient of automatic exposure control.

Remarks
This function retrieves the conditions set by CAM_AE_SetAdjust () function.

CAM_AE_SetSpotColor — Set AE spot drawing characteristics

```
void CAM_AE_SetSpotColor (bool bDrawingOnOff, PxRectEx  *pRectEx,
                          UCHAR red, UCHAR green, UCHAR blue)
```

Set the drawing characteristics of the AE spot rectangle.

Parameters
IN bDrawingOnOff - true if AE is visible, false if not visible
IN pRectEx - the rectangle specifying the outline of the spot indicator
IN red, green, blue - specify the color of the rectangle, with values between 0 and 255

Return Values
None

CAM_AE_GetSpotColor — Get AE spot drawing characteristics

```
void CAM_AE_GetSpotColor (bool bDrawingOnOff, PxRectEx  *pRectEx,
                          UCHAR red, UCHAR green, UCHAR blue)
```

Get the drawing characteristics of the AE spot rectangle.

Parameters
OUT bDrawingOnOff - true if AE is visible, false if not visible
OUT pRectEx - the rectangle specifying the outline of the spot indicator
OUT red, green, blue - specify the color of the rectangle, with values between 0 and 255

Return Values
None

CAM_AE_SetMicroMode — Set AE microsope mode

```
void CAM_AE_SetMicroMode (MicroModeEx     theMode)
```

Set the brightfield or flourescence microscope mode for calculating AE

Parameters
IN theMode - specifies the Microscope mode for calculating auto exposure

It can be any one of the following values.

Value	Meaning
kBF	Brightfield microscope illumination
kFL	Flourescence microscope illumination

Return Values
None

CAM_AE_GetMicroMode — Get AE microsope mode

```
void CAM_AE_GetMicroMode (MicroModeEx     *theMode)
```

Get the brightfield or flourescence microscope mode for calculating AE

Parameters

```
OUT   theMode - specifies the Microscope mode for calculating auto
            exposure

            It can be any one of the following values.
            Value             Meaning
            -------------------------------------------------------------
            kBF   Brightfield microscope illumination
            kFL   Flourescence microscope illumination
Return Values
      None
```

CAM_WB_SetMode **Set WB mode**

```
void CAM_WB_SetMode (WhiteBalanceModeEx Mode)


Sets the measurement mode of white balance.


Parameters
IN    Mode  Specifies the measurement mode of white balance.
            It can be any one of the following values.
            Value             Meaning
            -------------------------------------------------------------
      CAM_WB_AUTO       Measures white balance always.
            CAM_WB_1TIME      Measures white balance one time.
```

CAM_WB_GetMode **Get WB mode**

```
WhiteBalanceModeEx CAM_WB_GetMode (void)


Gets the measurement mode of white balance.

Parameters
      None

Return Values
      The return value is a measurement mode of white balance.
      See CAM_WB_SetMode().

Remarks
      This function retrieves the conditions set by CAM_WB_SetMode ()
function.
```

CAM_WB_SetRegion **Set WB region params**

```
void CAM_WB_SetRegion (bool bDrawingOnOff, PxRectEx  *lpRect,
                       UCHAR ucRed, UCHAR ucGreen, UCHAR ucBlue)

Sets the measurement region of white balance.
This region is only effective when the measurement mode is CAM_WB_1TIME.

Parameters
IN    bDrawingOnOff - Specifies the flag specifying if SDK draws the WB
                      region.
```

IN lpRect - Points to the PxRectEx structure that contains the measurement region of white balance. The position is specified on the movie image.
IN ucRed - Specifies the Red value of the frame for the white balance region.
IN ucBlue - Specifies the Blue value of the frame for the white balance region.
IN ucGreen - Specifies the Green value of the frame for the white balance region.

CAM_WB_GetRegion — Get WB region params

```
void CAM_WB_GetRegion (bool *pbDrawingOnOff, PxRectEx  *lpRect,
                       UCHAR* pRed, UCHAR* pGreen, UCHAR* pBlue)
```

Gets the measurement region of white balance.

Parameters
OUT pbDrawingOnOff - Retrieves the flag which specifies if the white balance region is drawn.
OUT lpRect Points to the buffer that receives the measurement region of white balance.
OUT pRed - Point to the valuable which retrieves the Red value of the frame color of white balance region.
OUT pGreen - Point to the valuable which retrieves the Green value of the frame color of white balance region.
OUT pBlue - Point to the valuable which retrieves the Blue value of the frame color of white balance region.

Return Values
None

Remarks
This function retrieves the conditions set by CAM_WB_SetRegion () function.

CAM_WB_SetFactors — Set WB factors

```
void CAM_WB_SetFactors (doube dRed, double dGreen, double dBlue)
```

Sets the white balance factors.

Parameters

IN	dRed	Specifies the red factor. It takes values from 0.0 to 2.0.
IN	dGreen	Specifies the green factor. It takes values from 0.0 to 2.0.
IN	dBlue	Specifies the blue factor. It takes values from 0.0 to 2.0.

Return Values
None

CAM_WB_GetFactors — Get WB factors

```
void CAM_WB_GetFactors (doube lpdRed, double lpdGreen, double lpdBlue)
```

Gets the current white balance factors.

Parameters

OUT	lpdRed	Points to the buffer that receives the red factor of white balance.

OUT	lpdGreen	Points to the buffer that receives the green factor of white balance.
OUT	lpdBlue	Points to the buffer that receives the blue factor of white balance.

Return Values
None

CAM_WB_CalibrateFactors — Calibrate WB

void CAM_WB_CalibrateFactors(void)

Measure the white balance, and set the white balance factors

Parameter
None
Return value
None

Remarks
This function cannot be used in automatic white balance mode.

CAM_BB_Start — Start black balance processing

void CAM_BB_Start(void)

Start black balance mode processing.
Parameters
None.

Return Values
None

CAM_BB_Stop — Stop black balance processing

void CAM_BB_Start(void)

Stop black balance mode processing.
Parameters
None.

Return Values
None

CAM_BB_SetRegion — Set BB region params

void CAM_BB_SetRegion(bool bDrawingOnOff, PxRectEx* lpRect,
UCHAR ucRed, UCHAR ucGreen, UCHAR ucBlue)

Sets the measurement region of black balance.

Parameters
IN bDrawingOnOff - Specifies the flag which specifies if SDK draws the black balance region.
IN lpRect Points to the PxRectEx structure that contains the measurement region of black balance. The position is specified on the movie image.

IN ucRed - Specifies the Red value of the frame of the black balance region.
IN ucBlue - Specifies the Blue value of the frame of the black balance region.
IN ucGreen - Specifies the Green value of the frame of the black balance region.

Return Values
None

CAM_BB_GetRegion — Get BB region params

void CAM_BB_GetRegion(bool *pDrawingOnOff, PxRectEx *pRC, UCHAR *pucRed, UCHAR *pucGreen, UCHAR *pucBlue);

Gets the measurement region of black balance.
Parameters
OUT bDrawingOnOff - Specifies the buffer retrieving the flag if the black balance region is drawn.
OUT lpRect Points to the buffer that receives the measurement region of black balance.
OUT ucRed - Specify a buffer retrieving the Red value of the frame for the black balance region.
OUT ucBlue - Specify a buffer retrieving the Blue value of the frame for the black balance region.
OUT ucGreen - Specifies a buffer retrieving the Green value of the frame for the black balance region.

Return Values
None

Remarks
This function retrieves the conditions set by CAM_BB_SetRegion () function.

CAM_BB_SetFactors — Set BB factors

void CAM_BB_SetFactors (double lpnRed, double lpnGreen, double lpnBlue)

Sets the black balance factors.

Parameters
IN lpnRed Red channel black factor level
IN lpnGreen Green channel black factor level
IN lpnBlue Blue channel black factor level

Return Values
None
Remarks
This function is used to set black level factors manually.

CAM_BB_GetFactors — Get BB factors

void CAM_BB_GetFactors (double *lpnRed, double *lpnGreen, double *lpnBlue)

Gets the black balance factors.

Parameters
OUT lpnRed Not used in monochrome SDK
OUT lpnGreen Points to the buffer that receives the monochrome factor of black balance.
OUT lpnBlue Not used in monocrhome SDK.

Return Values
None
Remarks
This function is used to retrieve the black level factors.

CAM_BB_CalibrateFactors — Calibrate BB factors

void CAM_BB_CalibrateFactors (void)

Measure the black balance, and sets the black balance factors.
Parameters
None

Return Value
None

CAM_EXP_SetSpeed — Set manual exposure

BOOL CAM_EXP_SetSpeed (double dExpSpeed)

Sets the manual exposure time to acquire still-image.
Exposure time is specified by the following formulas.
Exposure time = nExpSpeed / 1,000 [sec]
nExpSpeed

Parameters
IN dExpSpeed Specifies the exposure time to acquire still-image. It takes from 0.1 to 1,000.

Return Values
None

Remarks
This function can be used only in manual exposure mode.

CAM_EXP_GetSpeed — Get manual exposure

void CAM_EXP_GetSpeed (double *lpdExpSpeed)

Gets exposure time, for still-image and preview image.

Parameters
OUT lpdExpSpeed Points to the buffer that receives the exposure time.

Return Values
None

CAM_EXP_SetSensitivity — Set exposure sensitivity

BOOL CAM_EXP_SetSensitivity (int nISO)

Sets the ISO speed (sensitivity) to acquire still-image.

Parameters
nISO(IN) Specifies the ISO speed (sensitivity).

Return Values

TRUE - if successful
FALSE -otherwise.

Remarks
This function can not be used in AE-lock mode.

CAM_EXP_GetSensitivity **Get exposure sensitivity**

void CAM_EXP_GetSensitivity (int *lpnExpISO)

Gets ISO speed (sensitivity).

Parameters
OUT lpnExpISO Points to the buffer that receives the ISO sensitivity.

Return Values
None

CAM_IP_SetLuminanceLevels **Set luminance table parameters**

void CAM_IP_SetLuminanceLevels(LevelAdjEx *theLevels)

Set LevelAdjEx parameters for a specified color channel.

Parameters
IN theLevels Points to the LevelAdjEx structure that specifies parameters for the level adjustment table parameters. LevelAdjEx structure has the following form.

struct LevelAdjEx {
IN ChannelEx nChannel, can be any of the following values:

Value	Meaning
kLuminanceChannel,	overall luminance of monochrome channel
kRedChannel,	affects the red color channel, (not useful in monochrome camera mode)
kGreenChannel,	affects the green color channel, (not useful in monochrome camera mode)
kBlueChannel	affects the blue color channel, (not useful in monochrome camera mode)

IN int nInShadow, shadow level between 0 and 255
IN int nInHighlight,highlight level between 0 and 255
IN double nInGamma, gamma value of curve
IN int nOutShadow, not currently used,
IN int nOutHighlight, not currently used.
};

```
Return Values
      None
```

CAM_IP_GetLuminanceLevels **Get luminance table parameters**

```
void CAM_IP_GetLuminanceLevels(LevelAdjEx *theLevels)

Get LevelAdjEx parameters for a specified color channel.

Parameters
IN    theLevels   Points to the LevelAdjEx structure that specifies
            parameters for the level adjustment table parameters.

LevelAdjEx structure has the following form.

      struct LevelAdjEx {
      IN    ChannelEx   nChannel,   can be any of the following values:
                  Value             Meaning
                  ------------------------------------------------------
-----
                  kLuminanceChannel,values for the overall monochrome
                                          luminance channel
                  kRedChannel,            red channel level
                              (not useful in monochrome camera mode)
                  kGreenChannel,          green channel level
                              (not useful in monochrome camera mode)

                  kBlueChannel            blue channel level
                              (not useful in monochrome camera mode)
                  ------------------------------------------------------
-----

            OUT    int   nInShadow,  shadow level between 0 and 255
            OUT    int   nInHighlight,highlight level between 0 and 255
            OUT    double      nInGamma,   gamma value of curve
                   int   nOutShadow, not currently used,
                   int   nOutHighlight,    not currently used.
             };

Return Values
      None
```

CAM_IP_SetOrientation **Set image orientation**

```
void CAM_IP_SetOrientation(OrientationEx theOrientation)

Specifies motion and still image orientation transformation, if any.

Parameters
IN    theOrientation          may be one of the following values:

Value             Meaning                 ------------------------------
----------------------------
kNoTransform,     no image orientation transformation
kFlipHorizontal,  flip motion and still image horizontally
kFlipVertical,,   flip motion and still image vertically
```

```
kRotate180        rotate motion and still images 180 degrees
                  ------------------------------------------------------
-----
Return Values
      None
```

CAM_IP_GetOrientation — Get image orientation

```
OrientationEx CAM_IP_GetOrientation(void)
```

Returns motion and still image orientation transformation mode.

```
Parameters
      None

Return Values
Of type OrientationEx, may be one of the following values:

Value             Meaning                 ------------------------------
----------------------------
kNoTransform,     no image orientation transformation
kFlipHorizontal,  flip motion and still image horizontally
kFlipVertical,,   flip motion and still image vertically
kRotate180        rotate motion and still images 180 degrees
                  ------------------------------------------------------
-----
```

CAM_IP_ColorCapture — Set color capture mode

```
void CAM_IP_ColorCapture(ColorModeEx      theColorMode)
```

Sets color motion display and still capture mode for either full color or grayscale, or for monochrome camera.

```
Parameters
IN    theColorMode - specifies the color mode and type of camera used

            It can be any one of the following values.
            Value             Meaning
            ------------------------------------------------------------
      kMonochromeCam    Monochrome camera
            kColorCamMono     Color camera in monochrome mode
            kColorCamColor    Color camera in color mode
            ------------------------------------------------------------

Return Values
      None
```

CAM_IP_IsColorCapture — Get color capture mode

```
ColorModeEx CAM_IP_IsColorCapture(void)
```

Get color motion display and still capture mode: either full color or grayscale, or monochrome camera.

```
Parameters
      None
```

Return values
It can be any one of the following values.

Value	Meaning
kMonochromeCam	Monochrome camera
kColorCamMono	Color camera in monochrome mode
kColorCamColor	Color camera in color mode

CAM_FF_Start — Start focus finder

void CAM_FF_Start(void)

Starts focus finder processing for motion imaging.
Parameters
None

Return value
None

Remarks
This function resets the peak value of focus evaluation and evaluates focus parameters for every motion image processed.

CAM_FF_Stop — Stops focus finder

void CAM_FF_Start(void)

Stops focus finder processing for motion imaging.
Parameters
None

Return value
None

CAM_FF_IsRunning — Get focus finder mode

bool CAM_FF_IsRunning(void)

Return status (true or false) of focus finder processing for motion imaging.

Parameters
None

Return value
True if active, false if inactive

CAM_FF_GetEvalValue — Get last evaluation values of focus finder

void CAM_FF_GetEvalValue(int *lpnCurrent, int *lpnMax)

Get current and peak focus values for motion image stream.

Parameters
OUT lpnCurrnet points to the parameter that receives the current focus evaluaion value

OUT lpnMax points to the parameter that receives the maximum focus value

Return value
None

CAM_FF_Reset **Resets the maximum value of focus evaulation**

void CAM_FF_Start(void)

If focus finder is active, this function resets the maximum focus evaluation value to the current value..

Parameters
None

Return value
None

CAM_FF_SetRegion **Sets focus finder region**

void CAM_FF_SetRegion(bool bDrawingOnOff, PxRectEx *pRect, UCHAR red, CHAR green, UCHAR blue)

Sets focus finder rectangle size and position, visibility, and color.

Parameters
IN bDrawingOnOff Determines if focus finder rectangle is visible or not
IN *pRect Pointer to rectangle determining focus finder rectangle dimensions
IN red, green, blue Determine focus finder rectangle color

Return value
None

Remarks
Focus finder value is determined within the specified rectangle of the current motion image, whether rectangle is visible or not.

CAM_FF_GetRegion **Gets focus finder region**

void CAM_FF_GetRegion(bool *bDrawingOnOff, PxRectEx *pRect, UCHAR *red, CHAR *green, UCHAR *blue)

Gets current focus finder rectangle size and position, visibility, and color.

Parameters
OUT *bDrawingOnOff Pointer to boolean specifying whether focus finder rectangle is visible
OUT *pRect Pointer to rectangle specifying focus finder rectangle dimensions
OUT *red, *green, *blue Pointer to byte values specifying focus finder rectangle color

Return value
None

CAM_MIA_SetResolution **Set motion image resolution**

BOOL CAM_MIA_SetResolution (PreviewResolutionEx reso)

Sets the size of movie-image to acquire.

Parameters

IN reso Specifies the capture size of movie-image. It can be any one of the following values.

Value	Meaning
kFast,	FAST mode
kFull,	FULL mode
kZoom	ZOOM mode

Return Values

TRUE if successful
FALSE otherwise

CAM_MIA_GetResolution **Get motion image resolution**

void CAM_MIA_GetResolution (PreviewResolutionEx *pReso, PxSizeEx *pSize)

Gets the size of movie-image to acquire.

Parameters

OUT pReso Point to the buffer retrieving resolution.
OUT pSize Point to the buffer retrieving pixel size.

Return Values

None.

Remarks

This function retrieves the conditions set by CAM_MIA_SetResolution () function.

CAM_MIA_GetPixelSize — Get motion image pixel size

void CAM_MIA_GetPixelSize (double *lpdXSize, double *lpdYSize)

Gets the pixel size of movie-image. Unit of size is micron meter.

Parameters

OUT lpdXSize — Points to the buffer that receives the pixel size of horizon.

OUT lpdYSize — Points to the buffer that receives the pixel size of vertical.

Return Values

None

CAM_MIA_Start [obsolete] — Start motion image display

BOOL CAM_MIA_Start (HANDLE *phHandle)

Starts the sequence of movie-image acquisition.

Parameters

IN phHandle — Point to the image handle retrieving preview images.

Return Values

TRUE if successful

FALSE otherwise

CAM_MIA_StartAutoDraw — Start motion image display

BOOL CAM_MIA_StartAutoDraw (HANDLE *phHandle, HDC theHDC)

Starts the sequence of movie-image display.

Parameters

IN phHandle — Point to the image handle retrieving preview images.

IN theHDC — The current, safe drawing context

Return Values

TRUE if successful

FALSE otherwise

Remarks

Each frame will automatically be redrawn directly to the screen area above the window for the DC.

CAM_MIA_Stop — Stop motion image display

void CAM_MIA_Stop (void)

Stops the sequence of movie-image acquisition.

Parameters
None

Return Values

None.

CAM_MIA_IsRunning — Check if motion display is running

BOOL CAM_MIA_IsRunning (void)

Checks whether the sequence of movie-image acquisition is active.

Parameters
None

Return Values
TRUE Sequence of acquire the movie-image is running.
FALSE Sequence of acquire the movie-image is stopped.

CAM_MIA_Capture — Capture motion image

BOOL CAM_MIA_Capture (HANDLE *phImage)

Captures one frame of preview image.

Parameters
phImage - pointer to the handle receiving the captured preview image.

Return Values
TRUE - if successful
FALSE - otherwise

CAM_MIA_SetHDC — Sets motion image drawing context

bool CAM_MIA_SetHDC (HDC theHDC)

Sets the current HDC for the motion image drawing context

Parameters
theHDC - drawing context usually retrieved by calling GetDC()->GetSafeHdc()

Return Values
TRUE

CAM_SIA_SetResolution — Set still capture resolution

void CAM_SIA_SetResolution (StillResolutionEx resolution)

Sets the size of still-image to acquire.

Parameters
IN resolution
Specifies the capture size of still-image. It can be any one of the following values.

Value Meaning

---

kWarp,	Warp
kSingle,	Single
kVGA	VGA

Return Values
None

CAM_SIA_GetResolution **Get still capture resolution**

void CAM_SIA_GetResolution (StillResolutionEx* pReso, PxSizeEx* pSize)

Gets the size of still-image to acquire.

Parameters

pReso - Specify a buffer retrieving the resolution of still-capture image
pSize - Specify a buffer retrieving the pixel size of the still-capture image.

Return Values

None.

Remarks
This function retrieves the conditions set by CAM_SIA_SetResolution () function.

CAM_SIA_GetPixelSize **Get still capture pixel size**

void CAM_SIA_GetPixelSize (double *lpdXSize, double *lpdYSize)

Gets the pixel size of still-image. Unit of size is micron meter.

Parameters

OUT	lpdXSize	Points to the buffer that receives the pixel size of horizon.
OUT	lpdYSize	Points to the buffer that receives the pixel size of vertical.

Return Values
None

CAM_SIA_SetAccumulateTimes **Set still capture accumulation times**

void CAM_SIA_SetAccumulateTimes (int nTimes)

Sets the accumulate times to acquire the still-image.

Parameters
IN nTimes Specifies the accumulate times to acquire the still-image.
            It takes from 1 (no accumulate) to 64.

Return Values
    None

CAM_SIA_GetAccumulateTimes **Get still capture accumulation times**

void CAM_SIA_GetAccumulateTimes (int *nAccumulateTimes)

Gets the accumulate times to acquire the still-image.

Parameters

OUT The accumulate times to acquire an image.

Return Values

    None

Remarks
    This function retrieves the conditions set by CAM_SIA_SetAccumulateTimes () function.

CAM_SIA_SetAccumulateMode **Set still capture accumulation mode**

void CAM_SIA_SetAccumulateMode (AccumulateMethodEx Mode)

Sets the accumulate method.

Parameters
IN Mode
Specifies the accumulate method. It can be any one of the following values.

Value	Meaning
kNotAccumulating	
kAddition	Adds the image specified number of times.
kAddStopOnOverflow	Adds the image specified number of times. If detects a data of overflow, stops the accumulate.
kAverage	Averages the image specified number of times.

Return Values
    None
Remarks
    When accumulate method is kAddStopOnOverflow, monitor the region of photometry whether there is any data of overflow. kAddStopOnOverflow method is effective only at the case of auto exposure control. If the exposure state is otherwise (manual exposure, AE-lock), kAddStopOnOverflow method is same as kAddition mode.

CAM_SIA_GetAccumulateMode **Get still capture accumulation mode**

void CAM_SIA_GetAccumulateMode (AccumulateMethodEx *pMode)

Gets the accumulate method.

Parameters
OUT pMode The accumulate method to acquire an still-image.

Return Values
None

Remarks
This function retrieves the conditions set by CAM_SIA_SetAccumulateMode () function.

CAM_SIA_Start8Bit **Start 8 bit grayscale still capture**

BOOL CAM_SIA_Start8Bit (HANDLE *lphImage)

Starts the sequence which acquires the monochrome still-image of 8 bit length.

Parameters
OUT lphImage Points to the buffer that receives the memory handle of captured still-image.
Return Values
TRUE if successful
FALSE otherwise

CAM_SIA_Start16Bit **Start 16 bit grayscalestill capture**

BOOL CAM_SIA_Start16Bit (HANDLE *lphImage)

Starts the sequence which acquires the monochrome still-image of 16 bit length.

Parameters
OUT lphImage Points to the buffer that receives the memory handle of captured still-image.

Return Values
TRUE if successful
FALSE otherwise

CAM_SIA_Start24Bit **Start 24 bit still capture**

BOOL CAM_SIA_Start24Bit (HANDLE *lphImage)

Starts the sequence which acquires the still-image of 24 bit length.

Parameters
OUT lphImage Points to the buffer that receives the memory handle of captured still-image.
Return Values
TRUE if successful
FALSE otherwise

CAM_SIA_Start48Bit — Start 48 bit still capture

BOOL CAM_SIA_Start48Bit (HANDLE *lphImage)

Starts the sequence which acquires the still-image of 48 bit length.

Parameters
OUT lphImage Points to the buffer that receives the memory handle of captured still-image.

Return Values
TRUE if successful
FALSE otherwise

CAM_SIA_Prep8Bit — Prepare for 8 bit grayscale still capture

BOOL CAM_SIA_Prep8Bit (HANDLE *lphImage)

Prepares for the acquisition of a monochrome still-image of 8 bit length.

Parameters
OUT lphImage Points to the buffer that receives the memory handle of captured still-image.

Return Values
TRUE if successful
FALSE otherwise

Remarks
First phase of single or multi-capture sequence. Only needs to be called once, followed by any number of Capture sequences.

CAM_SIA_Start16Bit — Prepare for 16 bit grayscalestill capture

BOOL CAM_SIA_Prep16Bit (HANDLE *lphImage)

Prepares for the acquisition of a monochrome still-image of 16 bit length.

Parameters
OUT lphImage Points to the buffer that receives the memory handle of captured still-image.

Return Values
TRUE if successful
FALSE otherwise

Remarks
First phase of single or multi-capture sequence. Only needs to be called once, followed by any number of Capture sequences.

CAM_SIA_Start24Bit — Prepare for 24 bit still capture

BOOL CAM_SIA_Prep24Bit (HANDLE *lphImage)

Prepares for the acquisition of a still-image of 24 bit length.

Parameters

OUT lphImage Points to the buffer that receives the memory handle of captured still-image.

Return Values
TRUE if successful
FALSE otherwise

Remarks
First phase of single or multi-capture sequence. Only needs to be called once, followed by any number of Capture sequences.

CAM_SIA_Start48Bit **Prepare for 48 bit still capture**

BOOL CAM_SIA_Prep48Bit (HANDLE *lphImage)

Prepares for the acquisition of a still-image of 48 bit length.

Parameters
OUT lphImage Points to the buffer that receives the memory handle of captured still-image.

Return Values
TRUE if successful
FALSE otherwise

Remarks
First phase of single or multi-capture sequence. Only needs to be called once, followed by any number of Capture sequences.

CAM_SIA_CaptureExpose **Expose the still capture**

void CAM_SIA_CaptureExpose (void)

Exposure phase of the acquisition of a still-image.

Parameters
none
Return Values
none

Remarks
Exposure phase of single or multi-capture sequence. Sample may be illuminated immediately prior to this function call and light source darkened immediately afterward. CAM_SIA_PrepXXXX() must be called at least once prior to any number of expose and capture sequences.

CAM_SIA_CaptureProcess **Process the still capture**

void CAM_SIA_CaptureProcess (void)

Process phase of the acquisition of a still-image.

Parameters
none
Return Values
none

Remarks

Processes the exposed capture. CAM_SIA_CaptureExpose() may be called again immediately afterward, to capture another image of the same type, at the fastest possible framerate.

CAM_SIA_IsRunning — Status of capture process

```
BOOL CAM_SIA_IsRunning (void)
```

Returns the status of image still capture processing.

Parameters
None

Return Values
TRUE if capturing or processing still capture
FALSE otherwise

CAM_IMG_Save — Save image to specified format

```
int CAM_IMG_Save(ImageFileTypeEx  fileType, char* fullPathName, void*
          theImage)
```

Save image to specified type, with pathname

Parameters
IN fileType Specifies the accumulate method.
It can be any one of the following values.

Value	Meaning
kImageFileTIFF	TIFF image format
kImageFileBMP	BMP image format

IN fullPathName - pathname of file to save to
IN theImage - the image bitmap (see Remarks)

Return Values
Error code

Remarks
This function can only be used within the context of the WM_PIX_CAPTURE_DONE message sent by the SDK DLL after a still image capture is completed. The test application function "OnCaptureDone()" illustrates how to pass the image bitmap to this routine.